PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-113971
(43)Date of publication of application : 16.04.2002

(51)Int.Cl. B42C 19/00
B41F 23/00

(21)Application number : 2001-146168
(22)Date of filing : 16.05.2001

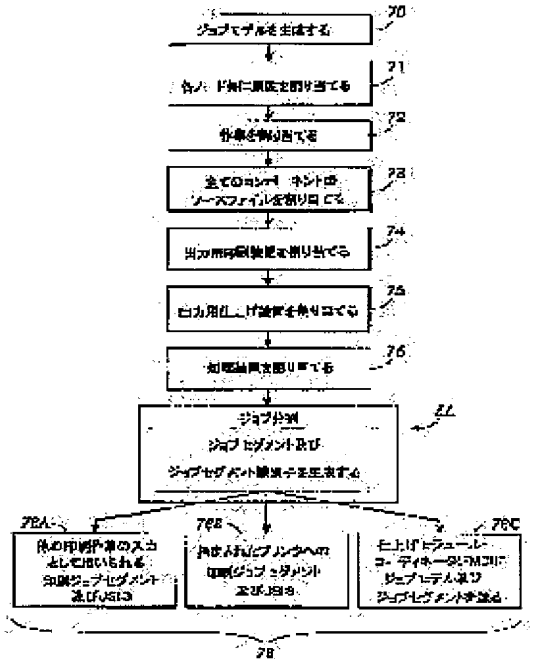
(71)Applicant : XEROX CORP
(72)Inventor : RYAN DONALD R
KREMERS HENRY T
MATHERS KEVIN R
SMITH WAYNE R
STURNICK GERARD R

(30)Priority

Priority number	Priority date	Priority country
2000 204460	16.05.2000	US
2000 204471	16.05.2000	US
2000 204624	16.05.2000	US
2000 204716	16.05.2000	US
2000 204720	16.05.2000	US

(54) APPARATUS AND METHOD FOR DESCRIBING, PLANNING AND AUTOMATICALLY PROGRAMMING COMPLEX FINISHING TASKS

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a system for the electronic management and control of a wide range of finishing processes characterized by input from multiple production operations and equipment that are variably applied to work objects that themselves are highly variable between different jobs.
SOLUTION: A job model file is created (70), attributes are assigned to each node identified in the high level job model (71) and the operations to be performed on each mode are identified. Source files for each of the document components are assigned (73), printers are assigned (74), output finishing devices are assigned (75) and processing devices are assigned (76). Job segments for operation of the job are determined based on the attributes and operations associated with each document component (77). Various outputs are shown (78A-78C).



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－113971
（P2002－113971A）
(43)公開日　平成14年４月16日(2002.4.16)

(51)Int.Cl.[7]	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｂ４２Ｃ　19/00		Ｂ４２Ｃ　19/00	２Ｃ０２０
Ｂ４１Ｆ　23/00		Ｂ４１Ｆ　23/00	

審査請求　未請求　請求項の数11　ＯＬ　外国語出願　（全121頁）

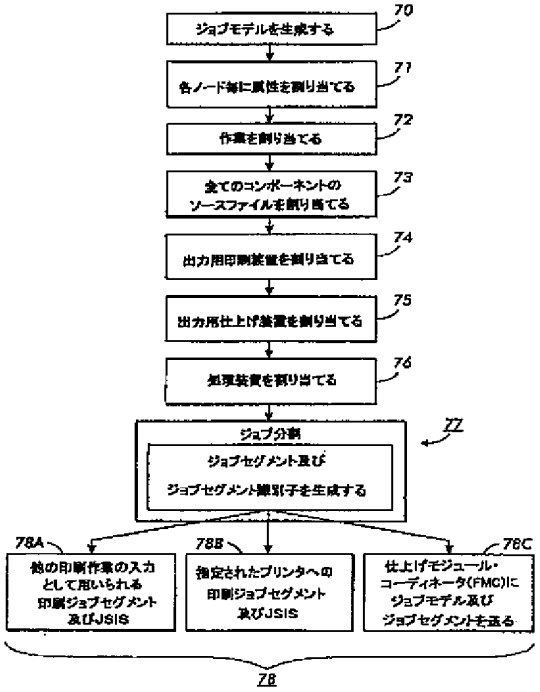
(21)出願番号　特願2001－146168(P2001－146168)
(22)出願日　平成13年５月16日(2001.5.16)
(31)優先権主張番号　２０４４６０
(32)優先日　平成12年５月16日(2000.5.16)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　２０４４７１
(32)優先日　平成12年５月16日(2000.5.16)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　２０４６２４
(32)優先日　平成12年５月16日(2000.5.16)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　590000798
ゼロックス・コーポレーション
アメリカ合衆国、コネチカット州、スタンフォード、ロング・リッジ・ロード　800
(72)発明者　ドナルド　アール．ライアン
アメリカ合衆国　14580　ニューヨーク州　ウェブスター　ブルー　クリーク　ドライブ　773
(74)代理人　100079049
弁理士　中島　淳　（外１名）

最終頁に続く

(54)【発明の名称】　複雑な仕上げ作業を記述し、計画し、自動的にプログラムするための装置及び方法

(57)【要約】
【課題】　それ自体様々なジョブ間で大きく変わり得る作用対象物に様々に変わる形で適用される多くの生産作業及び装置からの入力によって特徴づけられる広範な仕上げ工程の電子管理・制御のためのシステムを提供すること。
【解決手段】　ジョブモデルファイルが生成され（７０）、高水準ジョブモデルで特定された各ノードに属性が割り当てられ（７１）、各モードで実行される作業が特定される。各文書コンポーネント用のソースファイルが割り当てられ（７３）、プリンタが割り当てられ（７４）、出力仕上げ装置が割り当てられ（７５）、処理装置が割り当てられる（７６）。各文書コンポーネントに関連する属性及び作業に基づいてジョブの作業のためのジョブセグメントが決定される（７７）。種々の出力が呈示される（７８Ａ～７８Ｃ）。



【特許請求の範囲】

【請求項１】　ジョブの作用対象物を印刷し仕上げるための印刷・仕上げシステムにおいて：

ａ）ジョブの作用対象物を生成するための印刷装置と；

ｂ）該印刷装置とは別に制御され少なくとも１つの制約を有する、該印刷装置の出力を仕上げるための仕上げ装置と；

ｃ）少なくとも部分的に該仕上げ装置の制約に基づくジョブ調整情報を出力する生産監視コントローラと；

ｄ）該生産監視コントローラから出力されたジョブ調整情報を受け取った後、該仕上げ装置の動作を監督する仕上げモジュール・コーディネータと；

を備える印刷・仕上げシステム。

【請求項２】　上記生産監視コントローラが、少なくとも部分的に上記仕上げ装置の制約に基づいて決定されるジョブセグメントの識別情報を備えるジョブ調整情報を出力することを特徴とする請求項１記載の印刷・仕上げシステム。

【請求項３】　上記生産監視コントローラが、上記印刷装置によるジョブの少なくとも一部の印刷前に仕上げジョブセグメント情報の少なくとも一部を出力することを特徴とする請求項２記載の印刷・仕上げシステム。

【請求項４】　仕上げプロセスを統合し制御するためのシステムにおいて：

（ａ）製造プロセスで使用される装置の機能及び制約に基づいて生産ジョブをジョブセグメントに分離することができる生産監視コントローラと；

（ｂ）製造プロセスで使用される装置の機能及び制約に関する情報を記憶すると共にジョブセグメント記述を記憶するための少なくとも１つのデータベースと；

（ｃ）集成機／仕上げ機装置及び少なくとも１つのデータベースと通信して製造プロセス中にジョブセグメントを追跡するための仕上げモジュール・コーディネータと；

を備えるシステム。

【請求項５】　印刷ジョブの印刷と仕上げを調整する方法において：

ａ）少なくとも１つの制約を有する印刷装置を使用してジョブセグメントを印刷するステップと；

ｂ）印刷装置とは別に制御され少なくとも１つの制約を有する仕上げ装置を使用して、その印刷されたジョブセグメントを仕上げるステップと；

ｃ）少なくとも部分的に該仕上げ装置の制約に基づくジョブ調整情報を生産監視コントローラから出力するステップと；

ｄ）仕上げモジュール・コーディネータが該生産監視コントローラからジョブ調整情報を受け取った後該仕上げモジュール・コーディネータによって仕上げ装置の動作を監督するステップと；

を備える方法。

【請求項６】　製造装置とは別に制御される少なくとも１つの仕上げ装置を有すると共に、各仕上げ装置についての制約を含む装置依存パラメータ情報にアクセスし、ジョブの作用対象物の記述にアクセスし、該作用対象物が仕上げられる手法にアクセスすることができるコントローラを有する仕上げシステムにおける自動生産監視制御機能のための方法において：

（ａ）仕上げシステム内で使用するための少なくとも１つの仕上げ装置を選択するステップと；

（ｂ）その選択された装置の制約を特定するステップと；

（ｃ）その特定された制約の設定を各セグメントの属性が超えないようにしてコントローラで作用対象物の分離を指定するステップと；

を備える方法。

【請求項７】　上記（ａ）の仕上げ装置を選択するステップが、さらに、仕上げ作業を順次実行するための少なくとも２つの装置を選択するステップよりなり；上記（ｂ）の制約を特定するステップが、さらに、選択された装置の組合せに関連した一組の複合装置依存制約パラメータを作成するステップを備える；請求項６記載の方法。

【請求項８】　システム内で使用される装置の機能及び制約の属性に関する情報を記憶すると共にジョブセグメント記述情報を記憶するための少なくとも１つのデータベースを有し、かつジョブのコンポーネントが集成される順序と共にジョブのコンポーネントの記述を有する仕上げシステムにおける生産監視制御方法において：

（ａ）該少なくとも１つのデータベースから集成機／仕上げ機システム内で使用される装置の機能及び制約に関する情報を検索して取り出すステップと；

（ｂ）ジョブの処理のために集成機／仕上げ機システム内の少なくとも１つの装置を選択するステップと；

（ｃ）その選択された少なくとも１つの装置の複合制約属性を決定するステップと；

（ｄ）各セグメントの属性が該選択された少なくとも１つの装置の該複合制約属性を超得ないようにして、ジョブの作用対象物を分離するステップと；

を備える方法。

【請求項９】　システム内の装置の機能及び制約の属性に関する情報を記憶すると共にジョブセグメント記述情報を記憶し、かつジョブのコンポーネントが集成される順序と共にジョブのコンポーネントの記述を含むジョブモデルを記憶するための少なくとも１つのデータベースを有する仕上げ機システムにおける仕上げモジュール調整方法において：

（ａ）該少なくとも１つのデータベースからジョブセグメント及びジョブモデル情報を検索して取り出すステップと；

（ｂ）ジョブセグメントの状態を決定するステップと；

（ｃ）ジョブを処理するために使用される装置の状態を決定するステップと；
（ｄ）装置が動作するのに伴ってジョブの実行を監視するステップと；
を備える方法。
【請求項１０】　機能及び制約の属性を持つ仕上げ装置を有し、かつ該機能及び制約に基づいてジョブのジョブセグメントを決定すると共にジョブ及びそのコンポーネントの階層的記述を決定する生産監視コントローラを有する仕上げシステムにおけるデータベースシステムの方法において：
ａ）機能及び制約の属性をデータベース記憶するステップと；
ｂ）該機能及び制約の属性を該生産監視コントローラに伝えるステップと；
ｃ）ジョブセグメントを含めてジョブ及びそのコンポーネントの記述を記憶するためにデータベース内にジョブモデル記憶場所を設けるステップと；
ｄ）該生産監視コントローラジョブからジョブセグメントの記述を含めてジョブ及びそのコンポーネントについて記述した情報を受け取るステップと；
ｅ）データベース内の該ジョブモデル記憶場所にジョブセグメントを含めてジョブ及びそのコンポーネントの記述を記憶するステップと；
を備える方法。
【請求項１１】　データベース内の情報のノードとして記憶されるジョブ及びジョブセグメントの識別情報及び記述を含めて、ジョブのジョブモデルに関する情報を記憶するための少なくとも１つのデータベースを有する仕上げシステムにおけるグラフィカル・ユーザインタフェースにおいて：
ａ）データベース、該データベースに記憶されたジョブ及び該ジョブに関する情報を表示するための報告のタイプを選択するユーザオプションを備えた開始画面と；
ｂ）同じジョブ内の各ノードと他のノードとの関係を示すトリー・ビューとして配置された多数のノードを有する階層的ビューにおける選択されたジョブの任意形式のディスプレイと；
ｃ）ジョブ内の各ジョブセグメントの状態について記述した情報の任意形式のディスプレイと；
を備えるグラフィカル・ユーザインタフェース。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、それ自体様々なジョブ間で大きく変わり得る作用対象物にそのジョブによって様々に変わる形で適用される、多くの製造作業及び装置からの入力によって特徴づけられる広範囲な仕上げプロセスの電子管理及び制御に適用可能であると考えられる。本発明は、印刷及び印刷文書用の仕上げ作業との関連で説明するが、繊維製品の製造（捺染、裁断、縫製及び仕上げを含む）、様々な消費者向け製品及び産業用製品のパッケージング作業、プリント配線基板の製造等を含むような各種産業に応用可能である。ただし、本発明はこれらの産業に限定されるものではない。特に、本発明は、作用対象物の製造プロセスがそれらの作用対象物の仕上げ及びパッケージング作業のプロセスとは別個に管理されるような多くの作業に適用可能である。
【０００２】
【従来の技術】印刷文書の作成及び生産は、各ジョブ毎に大きく変わり得る多くの製造及び仕上げ作業を伴うことがしばしばある。一般に、それらの種々の作業は、大きく次のような３つのフェーズに分けることができる：１）文書を印刷に適した形で表現するプリプレス作業を含む文書情報の生成、２）紙のような何らかの形の媒体へのその文書情報の印刷、３）その選択された媒体の完成形の文書への仕上げ。これらの３つの主要フェーズはしばしば多くのサブフェーズを有し、プロセス全体は比較的単純な場合から非常に複雑な場合まで様々である。本発明は、ユーザが、当初の第１フェーズで既に、第３フェーズの終わりに到るまでの全プロセスを導くに足る十分な指示が得られるようにして、これらの３つの各フェーズ毎に詳細な指示を与える技術を提供するものである。本発明は、潜在的に多くの印刷作業で使用することができるが、マルチプリンタ、すなわちディジタルプリンタを使用する全面的なディジタル環境中で複雑な文書を作成し、印刷し、仕上げるための自動システムに特に好適に応用可能である。
【０００３】従来、フェーズ１で文書を作る際、その文書構成を行う者は作成しようとする文書の各部分を表す１つ以上の電子画像ファイルを作成する。これらの電子画像データファイルは、様々な文書生成・操作プログラムによって様々なフォーマットで保存することができる。例えば、表紙や絵・写真入り別丁にカラー印刷を用いる本のような複雑な文書については、ＰｏｓｔＳｃｒｉｐｔ（ポストスクリプト）（及びＰｏｓｔＳｃｒｉｐｔ互換言語のような種々のページ記述言語（ＰＤＬ）の中の任意の言語を用いてカラー画像を印刷可能な形式で表現することができる。多くの場合、文書中の異なるコンポーネントには異なるＰＤＬが利用されることになる。例えば、表紙は、異なる作業チームによって、あるいは写真リプリントまたは他の内部カラーコンポーネントと異なる装置で作成される場合がある。各プリプレスチームあるいはプリプレス装置は、それぞれの使用のために最適化されたＰＤＬを用いることが可能である。単純な白黒のテキストで構成されたページについては、デスクトップパブリッシング・プログラムを用いてそのようなページを表現することもできるし、あるいはより簡単なワードプロセッシング言語を使用することもできる。さらに、別丁、仕切ページ、及び完成形の文書の内部にあり得る他のコンポーネントの印刷については他の

プリプレス・フォーマットが用いられる場合もある。また、集成（製本）／仕上げジョブには、プラスチックセパレータ、保管棚（ストック品）から取り出された印刷済みシート、写真技術で制作されたシート、あるいはビニール・ディスクホルダや芳香サンプルパックのような特殊媒体等（これらに限定されるものではない）の非印刷コンポーネントが含まれる場合もある。

【０００４】ここで、コンポーネント及び複雑さのレベルが種々異なる文書のいくつかの例について図１及び２を参照して説明する。まず、図１には、丁合トレーまたは台上に下向きに載置された別丁コンポーネント１２、別丁コンポーネント１２の上に載置された中身コンポーネント１０、及びその後中身コンポーネント１０の上に載置された表紙１１を備える簡単な折丁文書が示されている。図中、符号Ｆ２０１のブロック形式で仕上げ作業が示されている。そのような仕上げ作業Ｆ２０１は、折丁中身の単純な折り作業のこともあれば、中央部ステープル留めあるいは同様の綴じないしは製本作業が含まれる場合もある。表紙１１が一番上に重ねられたならば、仕上げ作業２０１によって折丁が形成され、図示のように、表紙を綴じて製本された文書２１が得られる。完成した文書２１は、仕上げ作業Ｆ２０１の右側に示されている。説明の便宜上、各作用対象物の配置は仕上がった文書２１の下にボックス形式で示されている。

【０００５】図２は、２つの中身コンポーネント１０ａ及び１０ｂを２つの別丁コンポーネント１２ａ及び１２ｂと共に図示の順にスタック状に積層した結果を示したものである。表紙１１はこのスタックに最後に重ねられる。完成形の文書２３は、上記のようなスタックの構成から予期される９層からなり、中央の層は別丁コンポーネント１２ａを備える二重層になっている。

【０００６】文書は、コンポーネントの数と順序、選択した仕上げオプション等によって複雑さが大きく異ななり得るということは明らかである。通常、種々のプリプレス装置は、文書の個々のコンポーネントを作成し、それらのコンポーネントを印刷に適したフォーマットでディジタル的に表現する。このような目的のためには、Ｐｏｓｔｓｃｒｉｐｔ™ 一互換言語のようなＰＤＬがしばしば使用される。異なるプリプレスあるいは印刷作業が必要なジョブのサブセクションは、通常、プロセス初期の時点でオペレータによって分けられる。ジョブの各部分のプリプレス作業の終了後、オペレータは、そのジョブの様々な部分をそのような各部分に対応するプリンタに送ることにより、そのジョブの各部が取ることができるいろいろな「パス」を起動する。

【０００７】図３は、適度な複雑さの文書用の典型的な集成機／仕上げ機装置の動作を示す。図示例においては、カラープリンタによって１セットのカラー部分３０ａ、３０ｂ及び３０ｃが印刷され、プリンタから未校合のオフセット形式で出力された状態が示されている。また、１セットのモノクロ部分４０ａ及び４０ｂも印刷され、４０ａと４０ｂが交互に重ねられた校合済みオフセット・セットのスタックとしてプリンタから出力される。印刷及びそれぞれの中間出力ビンへの出力後、様々これら種々の印刷済みシートは、それぞれのプリンタ出力ビンから集められ、図３に示すビンに輸送して入れられ、集成機／仕上げ機装置に供給される。カラー部分３０ａ、３０ｂ及び３０ｃは、シートフィーダ４２の受け入れビン４２ａ、４２ｂ及び４２ｃに入れられる。製本装置と一体状に接続されたこのようなシートフィーダ装置の例としては、製本機モデルＳＰＦ－２０と一体化されたモデルＭＣ８０シートフィーダ（製本機、シートフィーダ共ホライズン・インターナショナル・インク．（Ｈｏｒｉｚｏｎ Ｉｎｔｅｒｎａｔｉｏｎａｌ，Ｉｎｃ．）社製）がある。モノクロ部分４０ａ及び４０ｂは、交互に重ねられた校合済みオフセット・スタックが維持されるようにしてセットフィーダ４３のフィーダビン４３ａに入れられる。このようなセットフィーダ装置４３の例としては、スタンダード・デュプリケーティング・マシンズ・コーポレーション，インク．（Ｓｔａｎｄａｒｄ Ｄｕｐｌｉｃａｔｉｎｇ Ｍａｃｈｉｎｅｓ Ｃｏｒｐｏｒａｔｉｏｎ，Ｉｎｃ．）社の販売になるドキュフィード１５０（ＤｏｃｕＦｅｅｄ １５０）がある。

【０００８】ここで、多くのジョブでは、４２ａ、４２ｂ、４２ｃ及び４３ａのような受け入れフィーダビンは、印刷されたジョブの特定の部分のスタック全高よりスタック高さを低くしなければならないという制約があるということに注目することが重要である。先行技術では、オペレータが、通常手動で、印刷されたシートのスタックを受け入れビンの高さの制約に適合するより小さいスタックに分ける。

【０００９】図３に戻って、校合装置４４は、コンポーネントを正しい順序に丁合・校合するようオペレータによってプログラムされる。運転時、校合装置４４は、シートフィーダ４２及びセットフィーダ４３と共に、丁合ステーションまたは丁合ビン４５の中で種々のシートが正確な順序で完了後のスタック５０中に置かれるよう動作する。次に、スタック５０は仕上げ機装置４６に送られ、そこで始めて折られる。その後、折られた折丁スタックは、綴じられ、化粧断ちその他の仕上げ作業を経て完成文書６０となる。仕上げ機４６で行われる仕上げ作業には次のようなものがある：あじろ綴じ（ｇｌｕｉｎｇ ｉｎ）、無線綴じ、平綴じ（ｇｅｎｅｒａｌ ｓｔｉｔｃｈｉｎｇ）、中綴じ、糸かがり、ぶっこ抜き、ステープル留め、折り目付け及び化粧断ち。

【００１０】多くの先行技術が、上記のような各々の装置及びプロセス内部の作業を自動化する処理や操作に取り組んでいる。特に、校合のような機能を備えた中間仕上げ部におけるプリンタからの出力を含め、プリプレス

作業とディジタル印刷プロセスとの間の自動的連係を達成するために多くの研究・開発の仕事がなされてきた。このような先行技術の一態様として、プリプレス装置から選択されたディジタルプリンタの中間仕上げ作業まで通して電子的に情報を伝達するための仮想ジョブチケットの作成がある。これに関しては、例えばローク（Ｒｏｕｒｋｅ）等に対して発行された米国特許第５，９９５，７２１号を参照すること。　このローク等の特許では、例えば、プリプレスプロセスで印刷ジョブの属性を調べることによって、種々ある印刷機器の中のどれがジョブの各特定部分を指定された属性に従って印刷することができるかが判断される。各特定の部分の印刷を管理する命令が仮想ジョブチケットに従って各プリンタに供給される。しかしながら、ローク等の特許及び他の先行技術においては、いろいろ異なるプリンタに送られる種々のジョブ部分のパス間のディジタル追跡・制御連係機能が、一般に、それらの各ジョブ部分が異なるプリンタに送られた後失われる。仮想ジョブチケットは、印刷プロセス自体、及びジョブの印刷フェーズに直接リンクされた何らかの印刷後プロセスの間にしか使用されない。その後、ジョブの分解部分は、オフライン仕上げへの指示を与える仮想ジョブチケットを用いてではなく、１つの分解部分のシートを第２の部分の印刷待ち行列に残っている「穴」に落とすことによって再統合される。これについては、ローク等の米国特許公報のカラム１３の１１乃至３９行目を参照すること。ローク等の特許及び他の先行技術に見られるもう一つの特徴は、印刷及び仕上げの全プロセスの間に遭遇する制約によってではなく印刷特性に基づいてジョブがいくつかの部分に分解されるということである。

【００１１】それぞれの中間仕上げ部に物理的に統合されたプリンタ内では２ウェイ方式のディジタル追跡・制御連係が一般化されているが、スタンドアロン型プリンタシステムと図３に４２～４６で示すようなオフライン集成機／仕上げ機装置との間には２ウェイ方式のディジタル追跡・制御連係機能はない。それぞれのプリンタと物理的に統合されていない集成機／仕上げ機システムに関しては、下記のようなジョブ及びジョブの作用対象物を特徴づけるデータの未記入リストが、一部または全ての集成／仕上げ作業をプログラムする際に有用である：各種類のシート数；媒体の種類；媒体厚さ；シートの向き；各シートのスタック内のシートの編成；シートまたはスタックを集成する順序；行うべき作業；折り目付け、折り、化粧断ち、断裁等のための場所；綴じの種類及び位置。このような指示及びパラメータの基本リストに加えて他の多くの指示もしばしば用いられる。

【００１２】先行技術中及び本発明の両方における必要は、効率的に上記の集成／仕上げデータを適切な集成機／仕上げ機システムに伝達しかつそれらのデータでその集成機／仕上げ機システムをプログラムし、その後仕上げ作業を通してジョブの進行状況を追跡し、最終的に欠陥仕上がり文書の発生を検知及び／または防止するためジョブの完全性を保つことである。

【００１３】先行技術では、上記の課題を様々な満足度で達成するいくつかの方法が教示されている。非常に一般的な取り組み方は、人間のオペレータが集成機／仕上げ機システムを個別にプログラムすることである。しばしば、仕上げ作業をプログラムする情報は、トラベラー・シートと呼ばれる手書きまたは印刷されたシートあるいはシートセットの形でオペレータに与えられ、トラベラー・シートにはジョブの全ての部分についての集成／仕上げ作業について記述した情報が組み込まれている。仕上げ装置にスタックをロードする準備をする段階で、オペレータは、コンポーネントを完成品に組む順序を含めて、トラベラー・シートを読んで適切な集成／仕上げに関する指示ないしは命令を得る。集成機／仕上げ機装置を準備し、プログラムする際には、オペレータは、自分の技量と経験によって適切なビンを選択し、スタックを分解し、シートを配向し、さらにはこれらと同様の作業を行うことができるので、トラベラー・シートでは各コンポーネントについての一群の属性が全て完全に与えられる訳ではない。これと同じ基本的システムのより新しい型のものでは、バーコードあるいは他の機械可読コード化情報でコード化されたトラベラー・シートが使用される。準備ができたならば、オペレータがトラベラー・シートをディジタルリーダにかけると、マニュアル・プログラミングのための情報がオペレータに表示される。この基本的な考えのもう一つの形は、各ジョブに関連付けられたシートにバーコードかグリフのような機械可読コードを設けるというものである。このシートは、例えば各シートスタック上に置かれたカバーシートであってもよい。コードはディジタルリーダによって読み取られ、その情報を用いて装置がそれぞれのジョブ用にセットアップされる。ある程度のジョブ情報をディジタル的にコード化する能力があっても、オフライン集成機／仕上げ機装置のプログラミングのための適度に複雑なジョブに関連して要求される情報は非常に複雑であるため、先行技術では何らかのマニュアル・プログラミングが要求される。

【００１４】上記の印刷、集成（製本）及び仕上げ作業の全てを相互に関連付けて自動的にプログラムする複雑さについては、かなり複雑な印刷ジョブ、特に多くのプリンタと多くの仕上げ作業及び集成機／仕上げ機装置が含まれる印刷ジョブの間に調整されなければならない夥しい数のパラメータ及び動作を考えることにより理解することができる。用紙選択（例えば寸法及び種類）に関連した全ての変数、ＰＤＬあるいは他のページ記述、及びプリプレス・フェーズ１から印刷フェーズ２の終了までに適用される他のパラメータ及び動作に加えて、集成／仕上げフェーズ３は、（１）そのジョブ内で各シート

あるいはシートセットに対して行われる集成及び仕上げ作業の複雑にわたる詳細を指定し、かつ（２）各シート及び各ジョブセットを他の全てのシート及びジョブセットと関連付ける情報のプログラミングが要求される。特にシートが、多くのプリンタから、あるいはプリンタと保管棚からというように異なる供給源からやって来る場合、既存のシステムは、これらのプログラミング・タスクの一部しか自動化することができず、あるいは非常に注意深く規定された状況でしかこれらのプログラミング・タスクを両方共実行することに成功してはいない。

【００１５】例えば、バリー（Ｂａｒｒｙ）等に発行された米国特許第５，８５９，７１１号では、この問題が、文書の全てのページが１つのジョブとしてジョブ・コントローラへ送られ、その後、各ページが個別の印刷ジョブとして扱われるようにページブレークで分割される特殊な場合について論じられている。１つのジョブをページ毎ベースで複数のジョブに分割することによって、種々のページを１台以上のプリンタを１台の仮想プリンタとして動作させることができるように多数のプリンタに割り当てることができる。その利点としては、カラーページをカラープリンタに、またモノクロページをモノクロプリンタに送るのに大きな速度と能力が得られ、そのためにこれらの各プリンタの使用が最適化されるということがある。上記のようにして、バリー等の特許は、印刷フェーズ２によってジョブの生産を最適化する方法を教示している。しかしながら、バリー等の特許は、生産の集成機／仕上げ機部分を最適化する方法、あるいは集成機／仕上げ機装置の機能及び制約を考慮した形でジョブの部分の印刷、分離、及びスタックを構成する方法を教示してはいない。バリー等の特許公報のコラム１８、３７～４７行には、ジョブが異なるプリンタに分解され再集成する必要がある場合発生する問題の解決手段の一部が下記のように記載されている：「唯一重要なのは、スタックが所与のプリンタ内で定義される場合、セパレータ間で生成された特定のスタックを所望の印刷ジョブ出力にある別のプリンタからの別のスタックと共に集成することを可能にするセパレータページのような何らかのしるしがあることである。」（４２～４７行）。また、コラム３７及び３８には、各スタック「を個別のエンティティとして扱い、待ち行列に並ばせ、所与のジョブ中の別のジョブスタックが隣接した印刷エンジンによってどれだけ速く処理されるかには無関係に処理するように」して、ジョブがどのようにスタックに分割されるかに応じて自動仕上げ装置を構成及び再構成し、分配機制御装置を動作させることができるということを教示したバリー等の解決手段の第２の部分が開示されている。（コラム３７、１２～１５行）。また、バリー等は、分布制御によって自動仕上げ装置を構成することができ、あるいはこのような仕上げ装置を各セパレータシート上にバーコードの形で印刷された命令を読むことによって構成することが可能なことを教示している。

【００１６】このように、バリー等の特許は、印刷作業をより効率的にするために異なるプリンタに回すことができる部分にジョブを分割する方法に関するものである。その仕上げに関する開示技術は、分離されたジョブの部分を最終仕上げのために再度正しい順序で再集成する方法に限られている。ジョブ部分を集成する方法に関しては、異なるスタックから供給される部分を校合する方法以外、全く何も教示されていない。印刷されたシートをストック品から取り出された、未印刷シートあるいは前に印刷されたシートと合わせて集成する方法については何も開示されていない。スタックを別々に分ける方法に関しては、印刷を最適化するために生成されたジョブ部分に応答して行うようにした技術を除き、何も教示されていない。集成／仕上げ作業を可能にするために印刷後にスタックのさらなる分割が必要な場合、バリー等の特許はこのような分割を分析しあるいは実施する方法についての記載がない。最も重要なことには、バリー等は、集成機／仕上げ機装置の機能及び制約を利用してジョブをいくつかの部分に分割できることを教示していない。

【００１７】恐らく、何らかの指定された形態の印刷ジョブの自動プログラミングのための体系を得ようという最も完全な試みは、フラウンホーファー・コンピュータグラフィックス研究所（Ｆｒｕｎｈｏｆｅｒ Ｉｎｓｔｉｔｕｔｅ ｆｏｒ Ｃｏｍｐｕｔｅｒ Ｇｒａｐｈｉｃｓ）によって発行されたプリプレス、プレス及びポストプレスの統合のための国際協力（Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｏｐｅｒａｔｉｏｎ ｆｏｒ Ｉｎｔｅｇｒａｔｉｏｎ ｏｆ Ｐｒｅｐｒｅｓｓ，Ｐｒｅｓｓ，ａｎｄ Ｐｏｓｔｐｒｅｓｓ）（ＣＩＰ３）プロダクション・フォーマット（Ｐｒｏｄｕｃｔｉｏｎ Ｆｏｒｍａｔ）である。ＣＩＰ３のウェブアドレスは次のとおりである：ｈｔｔｐ：／／ｗｗｗ．ｃｉｐ３．ｏｒｇ．１９９５年に初めて発行されＣＩＰ３は、Ｐｏｓｔｓｃｒｉｐｔ言語で書かれたディジタル「トラベラー・シート」を作成する手段を提供する。ＣＩＰ３によれば、文書及びその全ての生産ステップの完全なディジタル記述が可能になる。それは記述言語の提案であり、制御アルゴリズムではない。記述言語として、ＣＩＰ３は、ジョブに同じジョブの各ページごとに対して各ページを関連付ける１つの方法を提供する。ＣＩＰ３の規格に説明されているように、ＣＩＰ３フォーマットは集成機／仕上げ機装置の自動プログラミングを可能にすることを意図したものである。しかしながら、ＣＩＰ３に欠如しているのは自動プログラミング機能そのものである。また、記述された製造作業をこのジョブとの関連で利用可能な特定の装置の機能及び制約と合致させる機能も欠如している。従って、ＣＩＰ３は、ジョブを計画する場合に、使用される実際の装置を選択し、あるいは最適化す

るには有用ではない。また、ＣＩＰ３は、１つのジョブともう１つのジョブとの生産効果を相互に関連付ける機能を欠いており、従って、ＣＩＰ３は、多数のジョブの生産計画においては限られた価値しかない。また、ジョブを使用される実際の装置の制約に見合ったサブ部分に分割することができないことにより、ＣＩＰ３は、ジョブの各シートをそのジョブの一部として用意された他の全てのシートに関連付ける機能の深さという点でも限定されている。図４は、ＣＩＰ３を使用して記述することができる情報の形態を概略形式で示したものである。Ｙ軸に沿ってこの特定ジョブにおける種々のシートのリストが示されている。ここで、ＣＩＰ３は、生産ジョブを印刷するのに必要な手順の流れのシートにではなく、主に単一の最終製品のシートについて記述するものであるため、各タイプのについてどれだけ多くのシートが印刷されるかについては情報がないことがわかる。Ｘ軸は、各シートに対して行われる作業のリストになっており、この場合の動作には、中間の校合装置段階における種々のシートのスタックへの差し込みが含まれる。次に、これらのスタックは別個に折られ、断裁されてから、丁合され、化粧断ちされて綴じられる。ＣＩＰ３は、かなり複雑な製造プロセスを特定のシートあるいはシートのスタック（ＣＩＰ３内の部分製品と呼ばれる）と関連付けるのに十分な頑強性を有し、これらの製造プロセスにはジョブの多数の部分製品の丁合、このように丁合された部分製品を針金綴じあるいは接着綴じ文書として製本し、仕上げるプロセスが含まれる。ＣＩＰ３規格テーブル、バージョン３．０（１９９８年６月２日発行）の表３－１３を参照のこと。しかしながら、ＣＩＰ３でジョブの部分にとって適切な記述言語が提供される場合でも、生産時にジョブを実際に管理し、制御し、追跡するための方法は提供されない。言い換えると、ＣＩＰ３はジョブのプログラミングをその部分を記述することにより支援することを意図したものであるが、実際にはジョブのプログラミングの実施も行ったり、指図したりすることはない。

【００１８】複雑な印刷ジョブ中の各シートあるいはシートスタックに関連した複雑なデータを追跡し、関連付ける機能は、個々の装置の使用を特徴付け、制約する夥しい数の装置パラメータ及び処理パラメータによって大きく妨げられる。例えば、図３に示す各装置アイテムは、種々異なる用紙パス上の制約あるいはビン高さ上の制約を持つことが考えられる。いくつかのタイプのシートストックは、システム中のプリンタあるいは集成機／仕上げ機の一部にとっては薄すぎるかもしれず、あるいは硬すぎるかもしれない。ジョブのいくつかの部分は、ラミネーション加工のような中間仕上げステップが必要な場合もある。ラミネータのような特殊装置では、シートを特定の向き、例えば縦縁が先になる向きで送り、出力することが必要で、この制約のために、次の選択された装置の用紙パスにシートを挿入する前に向き変えプロセスが必要になる場合がある。ジョブのいくつかの部分が、その同じジョブの他の部分と異なる種類の折り、化粧断ちあるいは断裁作業を必要とする場合もある。典型的な市中の印刷所では、個々の異なる装置あるいは装置の組合せの選択に影響する制約の数および形態はそれぞれ固有で非常に多い。

【００１９】装置上の制約を調整する問題は、プリンタの出力と集成機／仕上げ機装置の制約との間の固有の不適合によって一層複雑化する。例えば、プリンタからのシートのスタックに集成機／仕上げ機のフィーダビンが受けることができるより多くのシートがある場合は、オペレータあるいは機械式運搬システムがフィーダビンの容量に適合すると思われるある数のシートをつかみ取ることによって、通常、その大きなスタックを分割する。このことは、いろいろな集成機／仕上げ機のフィーダビンが満杯になるシート数は同じではないということを意味する。また、それは、印刷されたシートのスタックは分割され、分離されるということを意味する。分離されたスタックは通常中間ビンに保存され、また、これらのスタックは必要に応じてそこに保存され、追跡され、そこから取り出されなければならない。シートの大きなスタックが、分離された各スタックが既知数のシートをもって維持されるようにマニュアルまたは自動計数手順で分割される場合であっても、スタックの大きさを各特定の印刷ジョブ及びそのジョブのために選択された特定の装置に影響を及ぼす変化する制約要因に合致するように動的に調整する機能は先行技術にはない。さらに、あらかじめ計数されたシートのスタックは、欠陥がある、欠落している、あるいは損傷していることが検知されたシートに対応する場合には、ほとんど役に立たない。最後に、ジョブを各々異なる制約を持つ様々な集成機／仕上げ機システムのうちの１つに回すことができる場合、選択されるに到ったその特定の集成機／仕上げ機装置の制約に最適に対応するために、印刷されたシートのスタックを動的に変えることが望ましいことも考えられる。このことは、装置破損あるいは初期選択された装置の使用のために利用不可であることに応じて集成機／仕上げ機作業の経路を指定しなおすことができるシステムで仕上げジョブの待ち行列を管理する場合は、特に望ましい。

【００２０】上に述べたように、印刷物製造プロセスの間に要求されるもう一つのタスクは効果的な追跡及び完全性の確認である。個々のプリンタシステム内の場合については、これらのタスクはよく理解されている。例えば、ゼロックス・コーポレーション（Ｘｅｒｏｘ　Ｃｏｒｐｏｒａｔｉｏｎ）社によって市場に出されているドキュテック（Ｄｏｃｕｔｅｃｈ）（モデル６１５５のような典型的な生産印刷システムでは、シートは、システムのインプット・フィーダから供給された時計数され、また、機械内で各主要動作が行われる間、またはそれら

の動作の後に計時されるか、あるいは再計数される。そして、シートがシステムによって許容される時間制限内に指定ステーションに到着しないか、他の何らかの原因で欠落していることが検知された場合、紙詰まりが宣言される、紙詰まり前に動作していた機械装置の部分が一時停止される。その後、オペレータは、用紙紙パス内にある全てのシートを取り除くことにより紙詰まり状態を解消するよう指示される。システム・コントローラは各シートを追跡し、計数しているので、取り除かれた各シートの番号及び画像ファイルがどのファイルであるかの情報を持っている。システムの全てのシートが取り除かれたことを検知すると、コントローラは取り除かれた各シートの再画像形成及び処理を指示する。再循環式文書フィーダ（ＲＤＦ）を備えた複写機システムについては、コントローラ最初の欠落した文書ページが画像作成待ち行列の一番上に戻るまで、インプット文書ページを循環させるようにＲＤＦに指示するという点を除いて、同じ考え方が当てはまる。

【００２１】多数のプリンタの使用が必要な印刷ジョブの場合、上記の追跡及び完全性確認機能は先行技術ではあまり十分には管理されない。例えば、１９９９年にゼロックス・コーポレーション社によって市場に投入された最先端技術のブック・ファクトリ（Ｂｏｏｋ　Ｆａｃｔｏｒｙ）（システムでは、ドキュテック・モデル（Ｄｏｃｕｔｅｃｈ　Ｍｏｄｅｌ）６１８０プリンタシステムに基づく一次生産プリンタ（ｐｒｉｍａｒｙ　ｐｒｏｄｕｃｔｉｏｎ　ｐｒｉｎｔｅｒ）が、次の仕上げ作業のうちのいずれかの作業とこれらの作業を適切に組み合わせた作業を行うことができる集成機／仕上げ機装置と物理的に統合されている：丁合、折り、化粧断ち及び無線綴じ。また、ブック・ファクトリ・システムは、他のプリンタあるいはプロセスによって印刷または作成された、表紙を含む保存棚からのシートを受け取るためのマニュアルロード式インプットトレーを備えた集成機も設けられている。ブック・ファクトリ集成機／仕上げ機装置は、単プリンタシステムとの関連で上に説明したのとほぼ同様にして進行するシートを計数、計時することができるが、手動でインプットされたシートを供給するプリンタまたはストックシステムとのジョブ状態に関する双方向通信機能は具備されていない。従って、紙詰まりが宣言された場合に、他のプリンタシステムに取り除かれたシートを自動的に印刷し直すよう指示することができる方法がない。通常の作業では、オペレータが、シートを刷り増しすることによってシートの欠落あるいは破損に備える。刷り増しコピーが利用可能できない場合は、オペレータは別のプリンタシステムをプログラムし直して、そのジョブを印刷し直さなければならない。いずれの場合も、通常刷り増しや機械から取り除いたシートの無駄が生じ、紙やインキのような消耗品が余分に消費され、さらにはオペレータ及び高価なプリンタによって貴重な時間が費やされる結果になる。

【００２２】

【発明が解決しようとする課題】要約すると、従来技術には下記のような機能あるいは特徴が欠けている：

１）プリプレス／印刷／仕上げプロセスの３つの全てのフェーズの対話型制御、追跡及び完全性確認機能のための統合型ディジタルアーキテクチャ；

２）ジョブの印刷前に、特にプリンタコントローラとオフラインである場合に、オペレータが複合した集成／仕上げ作業に完全な指示を与えることを可能ならしめる統合された装置及び方法；

３）プリンタコントローラ及び集成機／仕上げ機コントローラがどちらもそれぞれの印刷作業及び集成／仕上げ作業を指示し制御することができるような形で複雑な文書の最終文書様式を正確かつ完全に記述する方法；

４）利用可能な印刷システム及び利用可能な仕上げシステムの両方の制約に応じて印刷ジョブ及び関連するワークフローを分割し管理する装置及び方法；

５）指定されたプリンタ及び指定された集成機／仕上げ機装置の可用性に従って印刷・仕上げプロセス全体を管理する待ち行列管理システム；

６）各製造プロセスを通して、特にプリンタコントローラとオフラインの集成機／仕上げ機の作業を含むプロセス部分を通して各シートを追跡することができる対話型の完全性確認システム；

７）対話型の完全性確認機能に応じて機能し、集成機／仕上げ機装置がシートを要求してそれらのシートが得られない場合にプリンタにそれらのシートを刷り増すかあるいは交換するように自動的に指示ないしは命令する装置及び方法。

【００２３】

【用語の定義】本発明で使用する用語の定義は下記の通りである。

【００２４】「文書コンポーネント」とは、同様の特質あるいは特性を持ち、従って同様の仕方で印刷されるか、または未印刷のまま扱われ、また同様の仕方で仕上げられ、あるいは製造されるような１枚または２枚以上のシート媒体の集合を意味するものとする。文書コンポーネントの種類としては、例えば表紙、中身及び別丁（inserts）がある。ある特定の順序で一つにまとめられると、文書コンポーネントの集合は完全な「文書」を形成することが可能である。各文書コンポーネントは、最終的に「文書」に集成／仕上げされる前に独自の中間仕上げ作業が必要な場合がある。例えば、表紙は、最終集成／仕上げ装置に搬入する前の中間仕上げプロセスにおけるラミネーションが必要な場合がある。

【００２５】「文書」とは、特定の順序で置かれ、互いに種々の文書コンポーネントを関連付ける形で仕上げられた１つ以上の文書コンポーネントの集合を意味するものとする。

【００２６】「文書様式」とは、折り、断裁、綴じ、製本及び接着のような作業を含め、種々のコンポーネントが複合形式に仕上げられる仕方を意味するものとする。各文書様式は、それぞれ独自の画像組付け、印刷、仕上げプロセス要件及び物理的識別特性が必要である。最も典型的な文書の構造は下記の７つの標準「文書様式」のうちの１つに分類することができる：

ペラ：折られていない１枚以上のシート媒体で、綴じられていないものも、あるいはステープルまたは最初のシートの一方側から最後のシートの反対側へ媒体に通した針金のような手段によって綴じられたものも含まれる。

折丁：折られた１以上枚のシート媒体。シートは綴じられることも、綴じられないこともあるが、綴じられる場合は、折り目上で外側から内側へ向けて綴じられる。印刷されたシート媒体上には、（折り後に）シート上の画像が、読者が文書を開けた時読むのに正しい順序になっているページを形成するようにして、画像が組み付けられる。

無線綴（Ｐｅｒｆｅｃｔ　Ｂｏｕｎｄ）：一つにまとめられた、あるいは丁合済み折丁の個々のシート媒体を文書の中身／内容を保護する軟質のラップアラウンド表紙（薄表紙）に共通に接着剤で綴じ綴じ込んだもの。ブックブロックと表紙のエッジは通常互いに面一になっている。

くるみ製本（Ｃａｓｅ　Ｂｏｕｎｄ）：一つにまとめられた、あるいは丁合済み折丁の個々のシート媒体を文書の中身／内容を保護する硬質のラップアラウンド表紙に共通に接着剤で綴じ込んだもの。表紙のエッジは通常ブックブロックよりはみ出す。

レイフラット（Ｌａｙ　Ｆｌａｔｓ）：１つのエッジ沿いに穴または切り込みを設けた個々のシート媒体を一つにまとめ、中身／内容を保護する互いに別個の表紙と裏表紙に共通に綴じ込んだもの。共通綴じ込み方法としては、ワイヤ・コイル、プラスチック・コイル、プラスチック・コーム、３穴、５穴及び９穴リングバインダ等が用いられる。文書は、開かれると、平たく置かれ、ひとりでには閉じない。表紙のエッジは、通常ブックブロックと面一になっている。

テープ綴じ（Ｔａｐｅ　Ｂｏｕｎｄ）：個々のシート媒体を一つにまとめ、中身／内容を保護する表紙と裏表紙に共通に綴じ込んだもの。綴じ込み（製本）方法は、ブックブロックを接着テープで背表紙エッジ沿いに表紙及び裏表紙とオーバラップして包み込む。文書は、開かれた時、通常平たくはならない。表紙のエッジは、通常ブックブロックと面一になっている。必要に応じて、他の文書様式を指定することもできる。

【００２７】「制約」(constraint)とは、装置の設計あるいは使用に基づく何らかの制限を意味するものとする。「制約」は、永続的なものも一時的なものもある。永続的制約の例としては、変えられないビンの高さや幅、ラミネータの温度限界、ビンの種類、（セットフィーダかシートフィーダか）、送り方法（トップフィーダかボトムフィーダか）、要求された順序（ｎ→１か１→ｎか）、上向き、下向き、要求された向き（例えば、縦エッジ、横エッジのどちらを送り方向の前縁にするか）、紙パス幅、折り及び化粧断ちの厚さ、装置内部で可能な転換（例えば、上向きから下向きへの変更、前縁と後縁との逆転、風景写真の向きから人物写真の向きへの変更等）、及び装置設計に関連した同様の限界がある。一時的制約の例としては、次のようなものがある：（ａ）特定のビンのような１つの装置または装置の一部が部品の故障あるいは別のジョブに使用されていることにより利用できない期間；あるいは（ｂ）１つの特定の装置内で使用される媒体、接着剤、製本材料等の種類。

【００２８】「ジョブセグメント」とは、共通の印刷あるいは仕上げプロセスによって生産され、同じ印刷及び仕上げの制約に従うシートのスタックを意味するものとする。「ジョブセグメント」は、単一の文書コンポーネント、大きな文書コンポーネントの一部、あるいはいくつかの文書コンポーネントの校合が含まれ得る。以下に説明するように「ジョブセグメント」は、同様の印刷及び／または仕上げ要件を持つ文書コンポーネントが効率的な印刷、ハンドリング及び仕上げのために一つにまとめる目的で他と差別化される。例えば、文書が２つの８．５”×１１”サイズの白黒の中身コンポーネントを有する場合、その両方の中身コンポーネントは、同じプリンタ上で同時に印刷されるように同じジョブセグメントにまとめることができる。要求される条件ないしは要件によって、これらのコンポーネントはプリンタで校合済みまたは未校合スタックとして出力され、コンポーネントが校合される場合は、校合済みスタックは、校合済みセット間の区切りを指示するためにオフセット状に置くことが可能である。本発明と特に関連のあるもう一つの例として、選択された仕上げ装置のインプットビンが２．２インチのスタック高さ上の制約を持つ場合、「ジョブセグメント」の最大スタック高さは、ある特定の文書コンポーネントあるいは校合済みのコンポーネントのスタックのスタック全高がこれよりはるかに高くても、２．２インチになる。この状況の場合、印刷フェーズ２の間の「ジョブセグメント」は同じプリンタで印刷される全てのシートを備えることも可能である。しかしながら、この大きなジョブセグメント・スタック内では、高さ２．２インチに制限されたより小さな「ジョブセグメント」は、オフセット方式で分離することもできれば、セパレータシートによって分離すること可能である。従って、ジョブのセグメント区分は、他の点ではプリンタシステムの動作に影響することがないオフライン仕上げ上の制約に基づいて行われることになろう。

【００２９】本発明で使用される「仕上げ機」(finisher)及び「集成機／仕上げ機」(assembler/finisher)は、

いずれも集成及び／または仕上げ作業を行うよう設計されたシステムを指すものとする。

【００３０】「ノード」とは、単位の階層構造内に並べられたジョブの各単位、すなわちジョブ自体、ジョブ内の各文書、各文書内の各文書コンポーネント、各スタック、シート、シートセット等を意味する。ジョブの多くの「ノード」を識別あるいは特定するには、ＣＩＰ３に見られる階層記述子を効果的に用いることができる。オブジェクト指向ソフトウェアと共に使用する場合、「ノード」はそれ単独で移動または操作することができる「オブジェクト」として扱うことが可能であり、あるいはそれ自体が移動または操作することができるオブジェクトであるサブ部分の形に開くことも可能である。

【００３１】

【課題を解決するための手段】本発明の一態様は、複雑な文書の集成及び仕上げフェーズ３を、特定の順序で集成された時指定された文書様式のうちの１つに分類することができる一連の個々の文書コンポーネントとして文書が表されるアーキテクチャを使用することによって、ジョブの最初のフェーズ１のセットアップ時という早い段階で管理することができるソフトウェア・アーキテクチャにある。本願は、２０００年５月１６日付出願の仮米国特許出願第６０／２０４，６２４号により優先権を主張するものである。

【００３２】

【発明の実施の形態】ここで、図５によって、本発明のこの態様の概要を説明する。図５は、本発明を使用した仕事の流れを示しかつ本発明を使用する様々な装置アイテム間のいくつかの関係を示すブロック図である。図５において、ブロック１はフェーズ１のプリプレス作業を表している。ブロック１のプリプレス作業の出力は１セットの適切なＰＤＬファイルであり、生産モニタコントローラ（ＰｒｏｄｕｃｔｉｏｎＭｏｎｉｔｏｒ　Ｃｏｎｔｒｏｌｌｅｒ）（ＰＭＣ）１００に送られる。この後より詳しく説明するように、ＰＭＣは印刷ジョブの総合的生産を調整するコントローラである。

【００３３】図６は、ＰＭＣと仮想仕上げジョブチケット・データベース（ＶｉｒｔｕａｌＦｉｎｉｓｈｉｎｇ　Ｊｏｂ　Ｔｉｃｋｅｔ　Ｄａｔａｂａｓｅ）「（ＶＦＪＴＤＢ）５０１との関係を含めて、ＰＭＣ１００の典型的な入力及び出力をブロック図形式で示したもので、これについては以下にさらに詳しく説明する。一般に、ＰＭＣ１００の入力には、下記の一部または全部が含まれる：１）仮想プリンタ・ジョブチケット・データベース（Ｖｉｒｔｕａｌ　Ｐｒｉｎｔｅｒ　Ｊｏｂ　Ｔｉｃｋｅｔ　Ｄａｔａｂａｓｅ）（「ＶＰＪＴＤＢ」）（以下に説明する）のＶＪＴＤＢからのプリンタ機能(capability)及び制約のリスト；２）ＶＦＪＴＤＢからの集成機／仕上げ機機能及び制約のリスト；３）例えばＣＩＰ３による完成品の記述または同様の記述；４）印刷される各シートの内容のＰＤＬファイル及び他のファイル；５）印刷部数、ターゲット印刷装置、及び非印刷品目及び／またはストック品目の識別情報及び取り出し場所（これらに限定されるものではない）を含む特殊仕上あるいはパッケージング属性のような生産情報。一般に、ＰＭＣからの出力には、ジョブ内の各作業毎の各ジョブセグメントの識別情報、並びに各ジョブセグメント毎のフェーズ２の印刷及びフェーズ３の集成／仕上げに関する完全な命令セットが含まれる。より具体的に言うと、ＰＭＣからの出力には下記の一部または全てが含まれる：１）ジョブセグメントの記述及び各ジョブセグメントの識別子；２）ジョブセグメント、及びそのジョブセグメント内の文書コンポーネント、シートあるいはシートセットの構造のデータベース表現（以下に説明するＶＪＴＤＢ記述のような）；３）ジョブ・トラッキングシートがある場合はそのＰＤＬファイル；４）フェッチシート（ｆｅｔｃｈ　ｓｈｅｅｔ）がある場合はそのＰＤＬファイル；５）後で仕上げモジュール・コーディネータ（Ｆｉｎｉｓｈｉｎｇ　Ｍｏｄｕｌｅ　Ｃｏｏｒｄｉｎａｔｏｒ）（ＦＭＣ）で使用するためにコード化してＶＦＪＴＤＢに入れられた完全性記述子；６）プリンタ及び集成機／仕上げ機用の仮想ジョブチケット；７）１人以上のオペレータ応答を要求するプロンプト。ＰＭＣの内におけるプロセスについては、この後図７乃至１０を参照してより詳細に説明する。

【００３４】図５に戻って、フェーズ２の印刷用及びフェーズ３の集成／仕上げのための命令セットは、仮想印刷ジョブチケット（Ｖｉｒｔｕａｌ　Ｐｒｉｎｔ　ＪｏｂＴｉｃｋｅｔ）（ＶＰＪＴ）１０１及び仮想仕上げジョブチケット（Ｖｉｒｔｕａｌ　Ｆｉｎｉｓｈｉｎｇ　Ｊｏｂ　Ｔｉｃｋｅｔ）（ＶＦＪＴ）１０２の両方の形でＰＭＣ１００から出力される。ＶＦＪＴ及びＶＰＪＴは、ジョブ用の完全な命令セットを入れることが可能であり、あるいは単にこのような情報が保存されているデータベースへの参照ポインタだけを入れてもよい。ＶＰＪＴ１０１は、ロック等に対して発行された米国特許第５，９９５，７２１号に関連して前に説明したように当技術分野において従来周知であり；また、クリスト（Ｋｒｉｓｔ）等に対して発行された米国特許第５，６１５，０１５号；及びスクロット（Ｓｋｌｏｔ）に対して発行された米国特許第５，７６０，７７５号より当技術分野において従来周知である。ＶＦＪＴ１０２及びＰＭＣ１００によるその生成方法の詳細については、以下により詳細に説明する。

【００３５】各ＶＦＪＴのデータは、ＰＭＣによって図５に符号５０１で示す仮想仕上げジョブチケット・データベース（Ｖｉｒｔｕａｌ　Ｆｉｎｉｓｈｉｎｇ　ＪｏｂＴｉｃｋｅｔ　Ｄａｔａｂａｓｅ）（ＶＦＪＴＤＢ）に記録される。ＶＦＪＴＤＢは、印刷装置から送られて来る印刷物を受け取り、かつそれらの印刷物を所望の最

終文書様式にするのに必要な仕上げプロセスを実行するために要求される全てのジョブ構造データ、制御データ及び完全性データが入っているデータベースあるいはデータファイルである。ＶＦＪＴＤＢの様式は、ハードコピー（印刷物）、ソフトコピー（フロッピーディスク、ＣＤ－Ｒ、ＣＲ－ＲＷ）あるいは電子コピー（メモリまたはハードディスクドライブに電子的に保存する）のいずれの形であってもよい。また、人間可読型でも機械可読型でもよく、あるいは人間・機械可読型のものでもよい。

【００３６】各ジョブ毎にＶＦＪＴＤＢ５０１で要求されるデータ及び命令のタイプは、次のような情報である（ただし、これらに限定されない）：会計及び管理情報、シートレベル、セットレベル及びジョブレベルの仕上げ命令、カラー及び印刷品質の管理データ、登録等。また、これらのデータ及び命令には、生産中のジョブのジョブセグメント（スタック及びセットのスタック）の記述、及びどのようにこれらの部分を再び集成してジョブの処理を完遂するかについての命令も含まれる。さらに、この情報は、製造プロセスの十分なスコープの至る所での仕上げ装置、完全性管理及び監視の自動セットアップを可能にすることができる。ＶＦＴＤＢは、オフラインの仕上げ作業と、オンラインの印刷及び中間仕上げシステムの完全性管理機能、との間の直接連係のための基礎を与える。ＶＦＪＴＤＢデータは、独自のフォーマットあるいはＣＩＰ３の修正形式（これに限定されない）のような業界の標準フォーマットの形を取ることができる。ＶＦＪＴＤＢ５０１の構造及び使用に関するさらなる詳細については、以下に記載されている。

【００３７】次に、今図５に戻って、印刷プロセスのフェーズ２は、ＶＰＪＴ１０１によって、ブロック２００によって表されている１つ以上のディジタル・フロントエンド印刷コントローラ（Ｄｉｇｉｔａｌ　Ｆｒｏｎｔ　Ｅｎｄ　Ｐｒｉｎｔ　Ｃｏｎｔｒｏｌｌｅｒｓ）（ＤＦＥ）に渡された後、開始される。このようなＤＦＥは当技術分野において従来周知である。ＤＦＥの例としては、スプラッシュ（Ｓｐｌａｓｈ）社、ハーレクイン（Ｈａｒｌｅｑｕｉｎ）社、アドビ（Ａｄｏｂｅ）社他の製造になるＰＤＬ製品がある。ＶＰＪＴ１０１で与えられた命令に従って、印刷ジョブは別々の印刷ジョブセグメントに分割されて、種々の印刷エンジンに分配され、ＶＰＪＴが最初に確立された時オペレータまたはＰＭＣ１００が最適であると考えたプリンタあるいは印刷機を使用して印刷される。あるいは、場合によってはＰＭＣ１００との対話動作を通じてＤＦＥ２００が動的待ち行列・印刷物選択基準に基づいて自動的に適切な印刷装置を選択することが可能であることをＶＰＪＴに規定することもできる

【００３８】図５のブロック２０１～２０４は、文書コンポーネントを送入して印刷することができる様々な種類のプリンタの例である。プリンタ２０１は、統合型仕上げモジュール２０１Ａに接続されたカットシート・ディジタルプリンタである。プリンタ２０１と仕上げ機モジュール２０１Ａとの接続はＤＡＦあるいはＭＦＡタイプ・プロトコルを使用して行われる。上に述べたように、典型的な仕上げ機モジュール２０１Ａは、校合、折り、及びステープル留めのような簡単な製本のような機能を具備している。プリンタ２０２は、カラー印刷とモノクロ印刷機能を併せ持つカットシートプリンタである。このようなプリンタとしては、例えばゼロックス・コーポレーション社によって市場に出されているドキュメント・センター（Ｄｏｃｕｍｅｎｔ　Ｃｅｎｔｒｅ）（・カラーシリーズ（Ｃｏｌｏｒ　Ｓｅｒｉｅｓ）５０プリンタがある。仕上げ機モジュール２０２Ａは、図５に示すようにプリンタ２０２と統合され、仕上げ機２０１Ａに関して説明したのと同様の機能を持つことが可能である。同様に、プリンタ２０３は、図示の場合連続用紙フィードプリンタであり、仕上げ機モジュール２０３Ａと統合されている。プリンタ２０４は、通常オフセット印刷に関連した様々な装置及びプロセスを表し、これらの装置やプロセスにはプレートイメージャ２０４Ａ、プレート現像装置２０４Ｂ及びオフセット印刷機２０４Ｃでオフセット版を準備するプリプレスステップが含まれる。それぞれの中間仕上げモジュール２０１Ａ～２０３Ａとディジタル的に統合することが可能なプリンタ２０１～２０３と異なり、オフセット印刷機はディジタルイメージング装置ではなく、集成・仕上げ装置との直接的ディジタル統合はない。

【００３９】図５に示すように、２０１Ａ～２０３Ａの各仕上げモジュール及びオフセット印刷機２０４Ｃは、それぞれのジョブセグメントをそれぞれの出力トレーまたはビン２０１Ｂ～２０３Ｂ及び２０４Ｄに入れる。このようなトレーまたはビンに入れられた時、ジョブセグメントは、ハンドリング及び運搬できるように、校合され、スタック、あるいは他の形で分離される場合もされない場合もある。やはり上に述べたように、２０１Ａ～２０３Ａの各仕上げモジュールは、折りやステープル留めのような何らかの中間レベルの仕上げ機能を果たす。同じプリンタ及び中間仕上げ部で多数の文書コンポーネントを印刷し、あるいは集成することができ、ジョブのこのフェーズの間１つのジョブセグメントとして処理することが可能である。逆に、単一の大きな文書コンポーネントをその１つの文書コンポーネント内の複数のジョブセグメントを指示するセパレータシートあるいはオフセット・スタックによってスタックの形で出力することも可能である。

【００４０】本発明のもう一つの態様によれば、各ジョブセグメントに一意のジョブセグメント識別子（Ｊｏｂ　Ｓｅｇｍｅｎｔ　Ｉｄｅｎｔｉｆｉｅｒ）（ＪＳＩ）が関連付けられる。図５には、ＪＳＩが書き込まれたシート

がプリンタ２０１～２０４から出力される各ジョブセグメントと関連付けられる状態が示されている。各ＪＳＩシートには、それぞれ２０１Ｃ～２０３Ｃ及び２０４Ｅというラベルが付されている。複雑なジョブあるいは大きいスタックとして印刷された文書コンポーネントの場合は、ジョブ内あるいはスタック内の多くのジョブセグメントに対応して多くのＪＳＩを用いることが可能である。

【００４１】ＪＳＩは、仕上げ及び他の適用可能な印刷プロセス全体を通じてジョブセグメントと関連付けることができる任意の形式を取ることが可能である。このような形式としては、（ａ）印刷後、印刷されたジョブセグメントの一番上に置かれた印刷済みシート、（ｂ）ハードドライブのようなシステムメモリ、（ｃ）フロッピーディスクあるいは磁気ストリップのような磁気媒体、（ｄ）ＣＤ-ＲＯＭあるいはＣＲ-ＲＷディスクのような光学式メモリ、（ｅ）ジョブセグメントと関連付けられた、シート上に印刷されたバーコードシンボル、あるいは（ｆ）それによって機械または人間可読の識別情報がジョブセグメントと関連づけることが可能な他の何らかの手段の形で保存・記憶されたコピーがある。ＪＳＩは、ジョブのフェーズによって機械可読、人間可読、あるいは機械・人間共可読とすることが可能である。実際、スキャナーがジョブセグメントの一番上の印刷されたページをそのジョブセグメントを一意に特定することができるような形で読むことができる場合は、特殊な記号あるいは特殊なトップページは必要がなくなるはずである。従って、各ＪＳＩには、最小限、ジョブ番号及びジョブセグメント番号、あるいはジョブセグメントのみを他の全てのジョブセグメントから特定する他の識別子が入れられる。通常、ＪＳＩは唯一のジョブ番号と、ジョブセグメント識別子コード（Ｊｏｂ Ｓｅｇｍｅｎｔ Ｉｄｅｎｔｉｆｉｅｒ Ｃｏｄｅ）（ＪＳＩＣ）と、の両方を含む。ジョブ番号は印刷ジョブのみを他の全ての印刷ジョブから特定し、ＪＳＩＣはジョブセグメントのみを特定する。一実施例において、ＪＳＩＣはジョブセグメントの一番上のシート（以下トップシートとも称する）の上の特定可能な唯一のテキストよりなり、このＪＳＩＣはディジタルメモリにコード化されたまま保存されているＪＳＩへのベクトルを形成する。ＪＳＩがいずれの形式を取るにしても、ＪＳＩは特定されたジョブセグメントの内容を記述するＶＦＪＴＤＢの部分への参照ポインタとして機能する。ＪＳＩは、該当するジョブセグメントが印刷装置から他の仕上げプロセスに移送される時そのジョブセグメントと関連付けられた状態に保たれる。これは印刷装置から集成機／仕上げ機装置までの間にわたるジョブセグメントの追跡を可能にする。ジョブセグメントが印刷物が１つ以上の印刷装置で生成されることが必要なジョブの一部であるか否かにかかわらず、各ＪＳＩは共通ジョブ番号と、ジョブの各特定のジョブセグメントを一意に特定する異なるＪＳＩＣを持つことになる。

【００４２】図５に戻って、ＪＳＩは、通常ジョブのシートと共に印刷されて出力トレーあるいはビン２０１Ｂ～２０３Ｂ及び２０４Ｄのジョブセグメント・スタックの上に置かれるジョブセグメント識別子シート（Ｊｏｂ Ｓｅｇｍｅｎｔ Ｉｄｅｎｔｉｆｉｅｒ Ｓｈｅｅｔ）（ＪＳＩＳ）と呼ばれる印刷済みシートの形で示されている。このようなＪＳＩＳシートは、図５には２０１Ｃ～２０３Ｃ及び２０４Ｅで示されている。ＪＳＩＳについての情報は、（ａ）何らかの他の電子またはソフトコピー・フォーマットで保存された仮想仕上げジョブチケット・データベース（Ｖｉｒｔｕａｌ Ｆｉｎｉｓｈ Ｊｏｂ Ｔｉｃｋｅｔ Ｄａｔａｂａｓｅ）（ＶＦＪＴＤＢ）へのポインタ（ジョブ番号及びＪＳＩＣ）、あるいは（ｂ）ジョブのための命令を与えるＶＦＪＴＤＢ自体の部分を備える。このような命令は、電子可読あるいは人間可読の形式でＪＳＩＳ上に印刷することが可能である。スタックがどのように大きくても印刷済み出力の各スタックの上に置かれる従来のセパレータシートに対して、各ＪＳＩは印刷ジョブのそれぞれのジョブセグメントの唯一の識別子として機能する。ＪＳＩＳの一例が図７に示されている。ＪＳＩ及びジョブ命令を備える人間可読のテキストは部分６０１に示されている。部分６０２には、特定されたジョブセグメントに適用可能なＶＦＪＴＤＢの全データ内容を含む機械可読グリフ（glyphs）が示されている。部分６０３には、他の場所に保存されているＶＦＪＴＤＢへのポインタを備える機械可読バーコードが示される。ジョブの全てのシートと全てのＪＳＩＳが出力ビンまたはトレーに入れられたならば、印刷プロセスのフェーズ２は終了である。

【００４３】印刷プロセスのフェーズ３は、種々の文書コンポーネントが出力トレー２０１Ｂ～２０３Ｂ及び２０４Ｄから丁合され、特定の順序で集成され、指定された文書様式に仕上げられる最終の集成及び仕上げフェーズを備える。従来技術においては、多数の印刷装置が使われる場合、オペレータはフェーズ３の各ステップをフェーズ１及び２でそれぞれ行われる動作とは別に構成設定し、実行させる。集成及び仕上げが全て１台のデジタルプリンタ行われ、一つにまとめられた印刷／仕上げ機コントローラによって制御・管理される場合のみ、従来技術は、フェーズ３を自動的に構成設定し、制御することができるのということを教示している。図５で、矢印３０１及び３０２Ａ、Ｂ及びＣ（３０１Ａ、Ｂ及びＣ）は、出力トレーあるいはビン２０１Ｂ～２０３Ｂ及び２０４Ｄから仕上げのセットフィーダ・モジュール４０２及びシートフィーダ・モジュール４０１への印刷済みジョブセグメントの移送を示す。従来システムで、このような移送はしばしば手作業で行われるが、一部の用途では自動化された移送システムも用いられる。自動化され

た移送システムが用いられる場合でも、従来技術は、オフラインの集成機／仕上げ機装置をシート印刷前に生成された命令に基づいて自動的に複雑な集成及び仕上げ作業を処理するようにそれによってプログラムすることができる方法を教示してはいない。

【００４４】本発明においては、各ジョブセグメントはＪＳＩ参照ポインタと共に集成機／仕上げ機装置に到着する。上に述べたように、この参照ポインタは通常ＪＳＩＳ上に書かれるが、ＪＳＩは如何なる形式のものでもよい。ＪＳＩの目的は、集成機／仕上げ機の動作を制御する本発明のコントローラである仕上げモジュール・コーディネータ（Ｆｉｎｉｓｈｉｎｇ Ｍｏｄｕｌｅ Ｃｏｏｒｄｉｎａｔｏｒ）（ＦＭＣ）７００に対して特定のジョブセグメントを特定することである。図５において、仮想仕上げジョブチケット・リーダ（Ｖｉｒｔｕａｌ Ｆｉｎｉｓｈｉｎｇ Ｊｏｂ Ｔｉｃｋｅｔ Ｒｅａｄｅｒ）（ＶＦＪＴＲ）は符号７０１で示されており、ＪＳＩＳを読み取り、あるいはＦＭＣが唯一のＪＳＩＣを決定するのに十分な情報を他の何らかの形でＦＭＣ７００に供給する役割を有する。また、ＪＳＩＣをＦＭＣに渡すプロセスには、特にＪＳＩＳが人間しか読むことができない場合、人が介入してもよい。ＦＭＣ７００は、集成機／仕上げ機の生産データを管理し、解釈し、順番付け、割り当てるソフトウェアベースのコントローラである。各集成機／仕上げ機装置との種々のインタフェースを用いて、ＦＭＣはジョブの実行のために各装置をプログラムするのに必要なデータを各装置に伝達する。ＦＭＣは、プロセスを通して各ジョブセグメントを追跡して、装置が動作し始める前にジョブセグメントが正しくロードされることを確実ならしめる。また、ＦＭＣは、通常、オペレータに、ジョブ状態に関する情報、及び必要な場合あるいは適切な場合にオペレータが生産上の決定をすることができるようにするための情報を与える。ＦＭＣは、各ジョブセグメントを特定するＪＳＩを受け取り、かつＪＳＩ自身が必要な全ての集成機／仕上げ機データを書き込まれているかどうか判断することによって動作する。ＪＳＩＳあるいは同様のＪＳＩがジョブを仕上げるための命令を全ては与えない場合、ＦＭＣはＪＳＩＣを用いて、ＶＦＪＴＤＢに保存されたジョブモデルに関する全ての適切な情報を検索して取り出す。次に、ＦＭＣはＰＭＣによって用意された集成機／仕上げ機の組合せを調べて、特定された全ての装置がその時点で確実に利用可能であるようにする。この条件が満たされたならば、ＦＭＣは各ジョブセグメントが置かれるべきビンまたは他の集成機／仕上げ場所を決定する。一般に、ＦＭＣはＶＦＪＴＤＢを介してＰＭＣと通信する。集成機／仕上げ機装置が自動的にプログラムできる場合は、通常、ＦＭＣは、プログラミング命令を自動的に与えるために、各装置について指定されたインタフェース・フォーマットと対話動作するようにプログラムされる。また、ジョブ追跡及び完全性確認情報も供給されるようにすることが可能であろう。必要なジョブセグメントが全てのそれらの適切なビンにロードされたならば、ＦＭＣは集成機／仕上げ機装置に動作を開始するように命令するか、あるいはオペレータにジョブが実行可能であることを知らせるようにすることも可能であろう。このようにして、集成機／仕上げ機の動作全体を制御し、実行させ、追跡し、完全性を確認することができる。ＦＭＣの設計及び動作に関す詳細についてはこの後説明する。この発明の目的に関して言うと、ＰＭＣとＦＭＣの機能は互いに別個のコントローラ機能として説明されることに留意することが重要である。本発明においては、これらのコントローラを組み合わせることが可能であり、あるいは１つのコントローラとの関連で説明した何らかの機能を他のコントローラに改めて割り振ることも可能である。

【００４５】次に、図５にブロック１００で示すＰＭＣのについて、図８乃至１１を参照して詳細に説明する。図８は、本発明のＰＭＣの論理型アーキテクチャの一実施例の概要を図解したブロック図である。図８に示す実施例においては、あるジョブで使用される種々の装置の機能及び制約が、ＰＤＬファイルまたは同様の内容ファイルが生成された後におけるジョブを計画することを手伝うために、用いられる。本発明のもう一つの実施例では、機能及び制約に関するデータを、ＰＤＬファイル、組み付けファイル、及び同様の内容・レイアウトファイルを生成することを手伝うために、用いることができる。

【００４６】次に、図８に示す各ステップについて図９乃至１１を参照してさらに詳しく説明する。図８では、ステップ７０から始めて、ジョブモデル・ファイルを生成するプロセスが実行される。本発明を使用したジョブモデルの構築は、通常、ジョブについてのＰＤＬファイルあるいは他のページ記述ファイル、組み付けファイル及び同様のページ内容・レイアウトファイルの作成が終了した後、開始される。一実施例においては、ジョブモデルの構築は、ＣＩＰ３または同様のジョブレイアウト・フォーマットを用いてジョブがレイアウトされた後開始される。ステップ７０では、少なくともジョブ名、ジョブ識別子コード及びある識別子コードをデータベース中のファイルの場所に割り当てることによってジョブモデル・ファイルが開かれる。

【００４７】ステップ７１では、ＰＭＣが受け取った高水準ジョブモデルで特定された各「ノード」に自動的に、あるいはユーザーによって属性が割り当てられる。「ノード」は、前に定義したように、単位の階層構造中に並べられたジョブの各単位、すなわちジョブ自身、ジョブ中の各文書、各文書中の各文書コンポーネント、各スタック、シート、シートセット等を意味する。ＣＩＰ３に見られる階層記述子は、ジョブの多くの「ノード」

を特定する際に役立てることができる。オブジェクト指向ソフトウェアと関連して使用されるとき、「ノード」は、それだけで移動あるいは操作することができる「オブジェクト」として扱うことが可能であり、もしくはそれ自体が移動あるいは操作することができるオブジェクトであるサブ部分の形に開くことが可能である。ステップ７１で、特定可能なノードは通常ジョブ、文書及び文書コンポーネントのレベルである。

【００４８】ステップ７２では、各ノードで実行される作業が特定される。この場合も、ＣＩＰ３は何らかの作業の特定のための有用なツールとして用いることができる。次に、ステップ７３で、各文書コンポーネント用のソースファイルが割り当てられる。次に、ステップ７４において、ジョブを出力するためにアクセス可能なプリンタが割り当てられる。ステップ７５では、ジョブのために利用可能な集成機／仕上げ機装置が割り当てられる。ステップ７６では、ＰＭＣは校合のような中間の仕上げ作業をこのような機能を有するプリンタシステムに割り当てるべきかどうか、あるいはこのような作業は非統合型の集成機／仕上げ機装置によって行うべきかどうかの判断が行われる。ステップ７７では、ジョブセグメントを生成するという重要なステップが行われる。以下にさらに詳しく説明するように、ジョブの各作業のためのジョブセグメントは各文書コンポーネントに関連する属性及び作業及びこれに加えてそのジョブに使用することができる種々のプリンタ及び集成機／仕上げ機の機能及び制約に基づいて決定される。ステップ７８には、ＰＭＣからの種々の出力が示される。ステップ７８Ａには、中間ジョブセグメントの印刷及び／または他の準備のための命令が示されている。そのような中間ジョブセグメントの例は、最終的には２台以上のプリンタに通されることになるＪＳＩＳあるいは他のシートの最初の印刷であろう。このようなＪＳＩＣまたは他の中間準備の後、最初のジョブセグメントは、ばらばらに分解され、次の印刷あるいは集成機／仕上げ機のステップに適した別のジョブセグメントにまとめ込まれる。ステップ７８Ｂで、各ジョブセグメントの最終印刷のための命令が与えられる。ステップ７８Ｃで、ジョブモデル及びジョブセグメント記述のコピーがＶＦＪＴＤＢ（図５のブロック５０１）を介して仕上げモジュール・コーディネータ（ＦＭＣ）（図５のブロック７００）に送られる。７８Ａ及び７８Ｂの各ステップについては、通常、各ジョブセグメントに関連したジョブセグメント識別子シート（ＪＳＩＳ）または他のジョブセグメント識別子を生成するための命令が送られる。ステップ７８Ｃ内では、通常、ＪＳＩＳまたは他の識別子のコピーが直接あるいは間接的にＦＭＣに利用可能になる。

【００４９】次に、図９乃至１１を参照して図８の７０～７８の各ステップについてさらに詳しく説明する。図９には、ジョブモデルを生成する概ね図８のステップ７０～７３に対応するＰＭＣの部分の一実施例が示されている。図１０には、ジョブの実行に対して種々のプリンタ及び集成機／仕上げ機装置を割り当て、そのようなジョブ実行の動作を順序付ける概ね図８のステップ７４～７６に対応するＰＭＣの部分の一実施例が示されている。図１１には、各ジョブの毎にジョブセグメントを生成する概ね図８のステップ７７及び７８ａ～７８ｃに対応するＰＭＣの部分の一実施例が示されている。

【００５０】図９において、図示フローチャートのシーケンス動作は、ジョブの各シートの書類生成、操作及び組み付けに関わる全てのプリプレス・ステップを表すステップ８０で開始される。ステップ８０から得ることが可能な出力としては次のようなタイプのファイルがある：ＰＤＬファイル、シーニックソフト（Ｓｃｅｎｉｃ Ｓｏｆｔ，Ｉｎｃ．）社から入手可能なＰＲＥＰＳ組み付けソフトウェアによって生成されるような組み付けファイル、ＣＩＰ３ファイル、ゼロックス・コーポレーション社から入手可能なディジパス（ＤｉｇｉＰａｔｈ：登録商標）生産管理ソフトウェア、ワード（Ｗｏｒｄ）、パワーポイント（Ｐｏｗｅｒｐｏｉｎｔ）、フォトショップ（Ｐｈｏｔｏｓｈｏｐ）や他の任意プリプレス生成・作成ソフトウェアによって生成されるファイル。一般に、これらの各ファイルは、単一のシートまたは単一の文書コンポーネントに作用する。ステップ８１では、オペレータ（ユーザ）がジョブが保存されているデータベースを選択することによってジョブ生成プロセスを開始させ、そのようなデータベースに適用される規則に従ってジョブ名及びジョブ番号を割り当て、あるいは生成する。このデータはジョブノードを構成し、ステップ８１でこのようなジョブノードがデータベースに加えられる。ステップ８２では、ユーザが利用可能な文書様式のメニューから文書様式を選択し、このデータをデータベースのジョブノードファイルに入れられる文書ノードとして加える。ステップ８３では、ユーザが選択された文書様式に固有の属性のメニューから選択された文書様式属性を加える。これらの属性はデータベースの文書ノードファイルに加えられる。ステップ８４では、ユーザがデータベースの文書ノードより下位のレベルにファイルされるコンポーネント・ノードとして加えられるかまたは修正されるべき文書コンポーネントの名前を指定する。ステップ８５では、ユーザが特定のＰＤＬまたは他の内容ファイルをデータベースのコンポーネント・ノードと関連付ける。内容ファイルの例としては、ステップ８０との関連で上に説明したようなファイルが含まれる。ステップ８６では、各文書コンポーネントに適用できる属性が該当する文書コンポーネント・ノードに割り当てられる。この関連における「属性」とは、紙の種類や色のような記述子、組み付け情報、及びその特定の文書コンポーネントの処理に関するその他任意の記述的指示を意味する。ステップ８７で、ＰＭＣは、コンポーネ

ント記述子、ＰＤＬ及び属性全てが該当する文書様式に関する「文書コンポーネント様式規則」に合致するかどうかをチェックして確認する。例えば、折丁様式についての文書コンポーネント様式規則は、折丁のセンターシートが中央にのどアキのない２アップシートに関するＰＤＬと関連付けられることができる、ということを規定する場合がある。また、文書コンポーネント様式規則によって、折り目の位置が必ずのどアキ部に来るかどうかをチェックして確認することも可能である。ステップ８７のチェックで文書コンポーネント書式規則違反が検出されると、そのことがオペレータに通知され、プロセスはステップ８４に戻る。違反が検出されなかった場合は、ステップ８８で、各文書コンポーネントへの属性の割り当ては完全であると判断される。

【００５１】ステップ８９で、ユーザは、さらに他の文書コンポーネントがジョブに追加されるかどうかを尋ねられる。問い合わせの答えがイエスならば、ユーザインタフェースによってユーザはステップ８４に戻る。答えがノーならば、ＰＭＣアルゴリズムはステップ９０に進み、このステップで、「文書様式規則」演算子が全ての文書コンポーネントの属性を互いに、また選択された文書様式の中のコンポーネントに対して個別的に許されあるいは禁止される属性のリストと比較する。例えば、種々の署名の組み付け属性が互いに合致しない（例えば、のどアキと折り目の位置が合致しない）と、「文書様式規則」演算子９０が、ステップ９１で、その文書様式規則違反をユーザに通知し、オペレータによって規則違反が修正あるいは解消されるまでジョブモデルの完成を阻止する。ドキュメント及びその文書コンポーネントが全てそれぞれの文書様式規則に合致すれば、ジョブはステップ９２に移り、ここでそのドキュメントモデルは完全であると判断される。そうではなくて、文書様式規則違反が検出されたならば、ユーザは、ステップ８９に戻り、ステップ８４に戻ることにより適切な修正を行うよう求められる。ジョブがステップ９２に移ると、ユーザは、ステップ９３で、ジョブにさらなるドキュメントを加えるべきかどうかを尋ねられる。その答えがイエスならば、ユーザはステップ８２に戻ることになる。答えがノーならば、処理は続いてステップ９４に進み、そのジョブモデルは完全であると判断される。

【００５２】次に、図１０には、ジョブを実行するための装置及びシーケンスを選択するＰＭＣの部分が示されている。まず、ステップ１００では、図９に示すプロセスの終わりからジョブモデルを受け取り、自動的に、あるいはユーザ介入によってジョブデータベースにファイルする。一実施例においては、ジョブデータベースは、各々以下に定義されるような仮想印刷ジョブチケット・データベース（Ｖｉｒｔｕａｌ　Ｐｒｉｎｔ　Ｊｏｂ　Ｔｉｃｋｅｔ　Ｄａｔａｂａｓｅ）（ＶＰＪＴＤＢ）と仮想仕上げジョブチケット・データベース（Ｖｉｒｔｕａｌ　Ｆｉｎｉｓｈ　Ｊｏｂ　Ｔｉｃｋｅｔ　Ｄａｔａｂａｓｅ）（ＶＦＴＤＢ）の両方からなる仮想ジョブチケット・データベース（Ｖｉｒｔｕａｌ　Ｊｏｂ　Ｔｉｃｋｅｔ　Ｄａｔａｂａｓｅ）（ＶＪＴＤＢ）である。ステップ１０１では、プリンタ及び機能と制約のテーブルがＶＪＴＤＢから（好ましくはＶＪＴＤＢのＶＰＪＴＤＢ部分から）検索して取り出され、また集成機／仕上げ機の機能及び制約のテーブルがＶＪＴＤＢから（好ましくはＶＪＴＤＢのＶＦＪＴＤＢ部分から）検索して取り出される。一実施例においては、ＶＪＴＤＢには、アクセスすることができる全ての該当する装置に関連したデータが書き込まれているが、ＶＪＴＤＢは、ジョブの任意の特定の種類のための全ての装置の部分集合（一部）についてのデータのみを検索して取り出すようにセットアップすることも可能である。上記の構成によれば、ある特定の装置をある特定の種類のジョブ専用に割り当てることが可能になることに加えて、故障中、使用中、あるいは修理中の装置をそのような装置が利用不可能である時処理されているジョブとの関係で「オフライン」にすることが可能になる。ステップ１０２では、ＰＭＣは、ジョブモデルに保存された全てのジョブ属性を用いて、結果的に完成文書を作成することができると思われる印刷、集成及び仕上げ、すなわちＢ＆Ｗ（白黒）印刷、校合、折り、丁合等についての全ての包括的組合せ及びシーケンスをマップする。本願においては、これらの各々の組合せ及びシーケンスを「スレッド」と称する。ステップ１０３で、ＰＭＣはＶＪＴＤＢに記述されている装置の全ての機能及び制約についての検索して取り出されたリストを用いて、それらの検索された装置がそのジョブモデルで特定された作業及び属性を実行することができる全ての可能な個々のパスあるいはスレッドのリストを生成する。このようにして、図４に示すような包括的スレッドが個々に特定して特定された装置及びシーケンスを備える種々のパスまたはスレッド上にマップされる。ここで重要なのは、ＰＭＣはステップ１０３内の制約のリストを用いて、ジョブを処理する際必要になる種々のジョブセグメントを決定するということである。例えば、ある特定のスレッドがそのスレッド中の１つの装置における２．２インチのビン高さ制限に遭遇したとすると、そのスレッドを通って流れるジョブの部分が２．２インチのビン高さ制限に全て適合するようなジョブセグメントにマップされる。実際には、各スレッドまたはその部分におけるジョブセグメントはそのスレッドの全ての装置から導出される全ての制限を含む制約によって定義される。 より大きなフレキシビリティを得るために適切である場合は、ある装置を通過した後、ジョブセグメントをそれらのジョブセグメントを構成する文書コンポーネントに分解し、以後の１つ以上の作業で処理される別のジョブセグメントとして組み直すことも可能である。ステップ１０４で、ＰＭＣは、ジョブモデルとその

各ノードを調べて、ジョブの全ての文書コンポーネントが、種々の装置上の制約に全て合致しかつ解決不能な競合なしで機能させることができる特定の個々のスレッド上にマップすることができるかどうかを確認する。全ての文書コンポーネントをこのような方法で個々にマップすることができる場合は、ＰＭＣは直接ステップ１０７に進む。そうでない場合は、ステップ１０５で、ユーザは、そのような個々にマップすることができない文書コンポーネントを知らされ、（１）ＶＪＴＤＢにリストされた他の装置をこのジョブのために利用できるようにするか、あるいは（２）１つ以上の文書コンポーネントを修正するために図９に示すステップに戻るかのどちらかの機会を与えられる。ステップ１０６では、文書コンポーネントが修正される場合、ジョブモデルが修正される。 また、該当する場合、利用可能な装置のＶＪＴＤＢリストも修正される。

【００５３】ステップ１０７～１１０では、装置の使用競合をなくし、あるいはできるだけ少なくし、ジョブ実行のために最適なスレッド選択を決定するために、種々の可能なスレッドが互いに照合され、比較される。この最適化プロセスは、種々の装置に関する値及び優先順位を決定するための評価基準を与える任意の数のアルゴリズムをによって達成することができる。例えば、最適化プロセスがジョブ完成のための最短時間に基づいて決定される場合、そのような最適化のために可能なアルゴリズムでは、スレッドを通って流れる各シートまたはジョブセグメントに対して各装置によって行われるそれぞれの動作の期間を割り当てることになろう。そして、各スレッドに属する全ての期間の合計時間が計算され、最も短い生産時間をもたらすスレッドの組合せがスレッドの最適マッピングとして選択されることになろう。同様に、ジョブの推定コストを最適化するため、ある特定の装置の使用を最適化する、あるいは最小限にするため、同様の何らかの優先目標のため、あるいは最適化目標の任意の組合せのための最適化アルゴリズム及びＶＪＴＤＢデータを確立することができる。ステップ１０９では、スレッド内の種々のプリンタの中間仕上げ機能が比較され、最適化される最適化プロセスの特定部分が指示される。例えば、プリンタが校合済みの形、未校合の形、あるいは校合済みセットの形の２つの別々の文書コンポーネントを印刷することができる場合、この場合に可能なこれらの３つの形式には、異なる集成機／仕上げ機の動作が必要な３つの異なる可能なスレッドが伴う。そして、これらの異なるスレッドを互いに比較して最適化を達成することが可能である。複数の出力あるいは仕上げのオプション動作が可能な場合は常に、このような異なるスレッドを生成することが可能である。ステップ１１０では、もう１つの最適化プロセスの実施態様として、プリンタ及び／または仕上げの動作時に効果的に一つのジョブセグメントとしてまとめることができ、従って２つのサブ部分を持つ１つのノードとして扱われる文書コンポーネントがあれば、それはどの文書コンポーネントであるかが決定される。例えば、Ａ及びＢとして特定された２つのモノクロ文書コンポーネントを同じプリンタによって印刷することができる場合、これらをＡ、Ｂ；Ａ、Ｂ；Ａ、Ｂ；等の校合済みのオフセット状スタックの形で文書コンポーネントを出力する複合ジョブセグメントとして扱う１つのスレッドが存在し得る。その後これらのサブ部分ＡとＢが分離される場合は、これらのサブ部分もそれぞれのセット内でオフセット形式とすることが可能である。 分離された後、これらの文書コンポーネントＡ及びＢを他の文書セグメントと再び組み合わせることによって次の集成機／仕上げ機の動作に適用可能な新しいジョブセグメントが形成されるようにスレッドを設定することも可能である。ステップ１０７～１１０が終了すると、ＰＭＣは、ステップ１１１で、必要に応じてジョブセグメント、予測時間、予測コスト等の情報を含め、ユーザにスレッドの最適な選択に関する推奨を行う。図１２に示す一実施例においては、スレッドはＧＵＩの形でユーザに提示され、そのＧＵＩでは、（１）そのジョブのために利用可能な全ての装置の３次元表現の斜視図（透視図）、（２）各ジョブセグメントがその装置を通して取るであろうスレッドを示すマップ、及び（３）そのジョブを完成するために必要な種々のジョブセグメントの数及びシーケンスのようなデータが表示される。このようなジョブの流れの３次元表現は、好ましくは、図９に示すように、ジョブの計画時にも、実生産時にジョブを追跡するための手段としてもユーザが利用できるようにする。

【００５４】ステップ１１２～１１４では、ユーザはＰＭＣによって用意された推奨スレッドを検討して、承認するよう促される。推奨スレッドが承認されたならば、ジョブは図１１に示すＰＭＣの完全性確認及び追跡機能に進む。ステップ１３で推奨スレッドが承認されなければ、ユーザは、ステップ１１４で、推奨スレッドと異なるスレッドのマップを選択するオプションを与えられ、その場合ＰＭＣはステップ１０８に戻る。あるいは、ユーザは、利用可能な装置のリストに装置を加える、利用可能な装置の制約を修正する（例えばビン位置を動かしてスタック高さ制限を変える）、装置の制約に適合する別のジョブセグメントを手作業で作成する、あるいは同様の手動優先法を用いることによってさらなるスレッドを設計する道を選ぶことも可能である。また、このステップでは、先に使用されていたりあるいは保守の必要から利用できない装置をユーザの介入あるいは時間の経過によって利用可能な状態に戻すことも可能である。また、ユーザは、このステップで、ジョブ中の１つあるいはそれ以上の文書コンポーネントを修正するために図９に示すジョブモデル・プロセスに戻る（例えば、異なる温度の制約下でラミネータ装置を使用するために異なる

ラミネーション材を選択する）ことも可能である。ユーザが図１０に示すプロセスから得られるスレッドのマップを承認すると、ジョブは図１１に示すＰＭＣプロセスに移行する。

【００５５】図１１は、図１０に示すプロセスからジョブ分割情報及びマップ情報を受け取って開始される。ステップ２００では、完全性記述子が、図１０に示すプロセスに従って決定され、承認されたジョブセグメント中の各文書コンポーネント毎にコード化される。このような完全性記述子には、各文書コンポーネント毎に、文書コンポーネント中のシート数、印刷部数、及びジョブを追跡するのに役立つ、またジョブセグメントの各シートに関して各作業が終了したかどうかを確認するのに役立つような他の情報が含まれる。ステップ２００には、各々２つの別個のパスを備えるいくつかのサブステップが含まれる。パスＡは、特定のシートまたは文書コンポーネントのためのＰＤＬファイルが前に生成された完全性記述子なしでＰＭＣに到達する場合に使用される。ステップＡ２００Ａでは、ユーザは、このような完全性記述子中に含まれる属性あるいはパラメータを含めて、各文書コンポーネントに適用可能な完全性記述子のタイプを定義するよう促される。このような定義は、ＰＭＣに固定記述子を与え、一定範囲の記述子値を設定し、あるいはＰＭＣにファイルされているデータベースからＩＤＣを導出するように指示することが可能である。ステップＡ２００Ｂでは、完全記述子を保存する場所が選択される。ステップＡ２００Ｃでは、ステップＡ２００Ａでユーザによって完全性記述子が全く与えられなかった場合、ＰＭＣは、ステップＡ２００Ａでユーザによって与えられた情報及びステップＡ２００Ｂで特定された場所を用いて新しい完全記述子を生成する。ステップＡ２００Ｄでは、完全性記述子自身あるいは完全性記述子コード（Ｉｎｔｅｇｒｉｔｙ　Ｄｅｓｃｒｉｐｔｏｒ　Ｃｏｄｅ）（ＩＤＣ）が該当するＰＤＬファイルに加えられる。ＩＤＣは、完全記述子が格納される場所へのポインタを含む。ステップＡ２００Ｅでは、完全性記述子すなわちＩＤＣがジョブデータベース・ファイル中の適切なノードに加えられる。好ましくは、上に述べたように、このデータベースはＶＰＪＴＤＢ及びＶＦＪＴＤＢの両方を備えるＶＪＴＤＢである。

【００５６】あるいは、ステップ２００のＢカラムには、ＰＤＬファイルがＰＭＣに渡される前に完全記述子が作成される場合に適用されるプロセスが記述される。ステップＢ２００Ａでは、完全性記述子がＰＤＬファイルから読み出される。ステップＢ２００Ｂでは、完全性記述子がコードかされてＶＪＴＤＢあるいは他のデータベースに入れられる。

【００５７】ステップ２０１で、ＰＭＣはジョブモデルを用いて、文書コンポーネントがジョブを完成するためにストック品からワークフローに加えなければならないプレプリントされたシートまたは未印刷シートあるいは他の品目であるかどうかを判断する。このような品目としては、セパレータシート、製本材料、前に印刷された絵・写真入り別丁、香料含有芳香カード及び現在のジョブの一部としては印刷されない同様の品目がある。このようなストック材料のスタックは、通常このジョブのために用意されたラベラー・シートを持たないので、ＰＭＣは、ステップ２０１で、フェッチシートＰＤＬ（Ｆｅｔｃｈ　Ｓｈｅｅｔ　ＰＤＬ）を生成し、必要ならばその印刷を指示する。

【００５８】ステップ２０２では、ジョブセグメントのためのジョブセグメント識別子コード（Ｊｏｂ　Ｓｅｇｍｅｎｔ　Ｉｄｅｎｔｉｆｉｅｒ　Ｃｏｄｅ）（ＪＳＩＣ）が生成される。ＪＳＩＣは、フェーズ３でＦＭＣがＶＦＪＴＤＢ中のジョブセグメントの記述を検索することができるようにする唯一の識別子コードである。ＪＳＩＣは、ジョブ番号をジョブ中のスタック番号を表す連番状に生成された番号と結合することによって生成することができる。ＪＳＩＣ及びその使用に関しては、この後さらに詳細に説明する。ステップ２０３で、ＰＭＣは、必要に応じて、ジョブセグメント識別シート（Ｊｏｂ　Ｓｅｇｍｅｎｔ　Ｉｄｅｎｔｉｆｉｃａｔｉｏｎ　Ｓｈｅｅｔ）（ＪＳＩＳ）印刷用のＰＤＬファイルの生成を指示する。ＪＳＩＳには、ＪＳＩＣ、ジョブモデルから取得されたジョブを処理するための人間可読型指示シート、及び文書属性と文書コンポーネント属性のリストが含まれる。しかしながら、ＪＳＩＳの印刷なしにジョブセグメントの識別情報を確認しあるいは追跡することができるシステムにおいては、ステップ２０３の一部または全部を省略できるということに注意するべきである。例えば、ＶＪＴＲがスタックのトップシートの内容を自動的に読むことができ、予測されるトップシートのデータベースと照合することができる場合は、ＦＭＣは、種々の生産フェーズを通って移動するそのようなトップシートを読むことによって、特別なＪＳＩＳの必要なしに、あるいはトップシート上に特別なＪＳＩＣを書く必要もなしに、ジョブセグメントを特定し、追跡することができよう。すなわち、トップシート自体の内容がＪＳＩを形成することになる。

【００５９】ステップ２０４で、ＰＭＣは、ＪＳＩＣ、ジョブ分離及び選択されたジョブセグメントスレッド情報をＶＪＴＤＢに記憶する。ステップ２０５では、ＪＳＩＳがある場合、それを扱うＰＤＬファイルが該当するジョブセグメントの印刷に必要なＰＤＬ及び他のファイルに加えられ、それらのファイルが該当するプリンタに渡される。これによって印刷製造プロセスのフェーズ２が開始される。ステップ２０６では、ジョブについてのフェッチシートもジョブの生産につれてオペレータが利用できるように該当するプリンタに送られる。ステップ２０７で、ＰＭＣは、そのジョブにまだ処理するべき他

のジョブセグメントがあるかどうかを判断する。判断結果がイエスならば、それらの各ジョブセグメント毎にステップ２００から２０６が再度繰り返される。全てのジョブセグメントの処理が終了したならば、ＰＭＣは、ステップ２０８で、そのジョブ及びその各ジョブセグメント及び／または各文書コンポーネントを生成して処理するのに必要な全ての情報をオペレータ用に人間可読形式にするようにトラベラー・シートＰＤＬの生成及びその印刷を指示する。そのようなトラベラー・シートは、通常、印刷所内でジョブジャケットと共にあちこち移動する。

【００６０】次に、図１３を参照しつつＶＪＴＤＢについてさらに詳細に説明する。特に、ＶＪＴＤＢのＶＦＪＴＤＢ部分についてより詳しく説明する。上に述べたように、ＶＦＪＴＤＢは、印刷装置から送られてくる印刷物を受け取り、それらの印刷物を所望の最終文書様式にするのに必要な仕上げプロセスを実行するために要求される全てのジョブ構造データ、制御データ及び完全性データが入っているデータベースあるいはデータファイルである。そのため、ＶＦＪＴＤＢあるいは他のアクセス可能なデータベースの一部には、シートの印刷が行われている時生じる動作のためにアクセス可能な全ての装置のテーブルまたはリストが、各装置の可用性及び優先順位パラメータや各装置の機能及び制約のリスト（ただしこれらに限定されない）を含めて入れられる。ＶＦＪＴＤＢの別の部分には、各特定のジョブに関するデータが入れられる。

【００６１】ＶＦＪＴＤＢのジョブ記述部分は、トリーの各レベルに１つ以上の別個のノードがある階層木構造を有するデータベースとして考えると最もよく理解される。図１３は、折丁文書の生成に当てはめられるようなトリー構造を示したものである。最も上のレベルＳｉｇ１は基本的なジョブ記述を表す。このノード内でコード化されるのは、ジョブ名及びジョブ番号、ノード識別子（この場合はＳｉｇ１）、顧客に関する情報、経理及び請求書発行情報、受注日、生産予定日付及び引渡し予定期日のような日程計画データ等の詳細情報である。これら及びその他の詳細情報はノードＳｉｇ１と関連付けられた「属性」であり、ノード自身の中にテーブル形式で保存するか、あるいは外部データベースへのＳｉｇ１の内の参照ポインタによって関連付けることができる。また、ノードＳｉｇ１内には、通常、任意の「親」ノード及びＳｉｇ１のすぐ下のレベルのノードへの参照ポインタがある。特定のノードのすぐ上及びすぐ下ノードのノードのこのような参照ポインタは、このＶＦＪＴＤＢの実施例の全てのレベルで特徴をなしている。このようなポインタの跡をたどることによって、そのような１つのノードが特定されれば、ジョブ全体を再集成することができる。

【００６２】Ｓｉｇ１の一つ下位のレベルには、特定の文書がジョブＳｉｇ１の一部として作成されるべき記述子を含むノードであるノードＰ１がある。特定のジョブの下に入れることが可能な文書の数には制限はなく、例えばノードＰ２、Ｐ３等が存在し得る。ノードＰ１に戻って、このような一番上のレベルの文書ノードには、文書を管理する文書様式の識別子が入っている。このような文書様式を指定することによって、そのような文書様式についての属性のテーブル及び１セットの文書様式規則（Ｄｏｃｕｍｅｎｔ　Ｆｏｒｍ　Ｒｕｌｅｓ）がＰ１と関連付けられ、これらの規則及び属性ジョブの生産をマップする際にＰＭＣによって使用される。また、上述べたように、ノードＰ１はその各子ノードも特定する。ここで重要な一つの態様は、各ノードＰ１には、子ノードの各々を互いに関連付ける順番及び方法を詳しく記述した情報、例えば表紙Ｃ１は最終的に文書の外側に置かれる文書コンポーネントであると確認されるような情報が入れられるということである。図１３に示す例は折丁小冊子であるから、Ｐ１の子ノード間の関係は、それぞれの場合に応じて各文書コンポーネントがその折り及び／または製本（綴じ）の前に丁合済み折丁スタックに加えられる順序によって指定することができる。また、ＰＭＣがジョブの生産に関連するスレッドについて記述するデータを保存することができるジョブモデル・レジスタもノードＰ１に入れられる、あるいは参照される。このようなレジスタには、全桁書き込まれた状態で、文書Ｐ１の「ビルド・シーケンス」のような詳細情報が記憶されることになる。図示ジョブの場合のビルド・シーケンスの一例は例えば下記のようなものであろう：１．プリンタ＃４のビン＃１から１枚のジョブセグメントＣ１を送り、集成機／仕上げ機（Ａ／Ｆ）＃２のビン＃６にそのシートを入れる。２．セットフィーダ＃１のビン＃１にあるジョブセグメントＢ１から最初のオフセット・セット（ジョブセグメント・スタック中の１つの文書コンポーネントを備える）を送り、そのセットをＡ／Ｆ＃２のビン＃６内の一番上に置く。３．ストック品移送システム＃１のビン＃２０からジョブセグメントＩ１（別丁を備える）を送り、そのセットを９０°だけ回転させて向きを変え、別丁Ｉ１をＡ／Ｆ＃２のビン＃６内の一番上に置く。４．ジョブセグメントＢ１から２番目のオフセット・セット（ジョブセグメント・スタック中の２番目の文書コンポーネントを備える）を送り、その２番目のセットをＡ／Ｆ＃２のビン＃６内の一番上に置く。５．ストック品移送システム＃１のビン＃５からジョブセグメントＩ２（２番目の別丁を備える）を送り、その別丁Ｉ２をＡ／Ｆ＃２のビン＃６中の一番上に置く。

【００６３】また、ジョブモデル・レジスタには、該当する文書様式規則に従って、折り目付け、折り、スティッチング、化粧断ち等のノードＰ１に記述された折丁小冊子の最終生産に帰着する全ての作業を実行するようにＡ／Ｆ＃６をプログラムするための情報も書き込まれ

る。このような作業のためにコード化される典型的なパラメータとしては、作業の識別情報と作用対象物上におけるその作業の位置、何らかのラミネーション・ステップの種類と温度、折り目付け、折り、製本等の何らかの圧力要件、綴じ種類と位置等がある。

【００６４】ノードＰ１より下位レベルの各ノードには、そのようなノードによって表されるジョブセグメントに関連する情報が含まれる。このような情報としては、そのようなジョブセグメントが取り出されるプリンタ及びビン、ジョブセグメント中のセット数またはシート数、その現在位置、校合済みか未校合か、上向きか下向きか、方向、シートの重量、種類及び厚さ等のセグメントの属性が含まれる。図示例で特に注目されるのは、交互状のオフセット・セットとして並べられた２つの校合済み文書コンポーネントからなる単一のジョブセグメントを備えるノードＢ１である。これらの各文書コンポーネントは、ノードＳ１で示されているようなそれぞれ自身の別ノードに記述することが可能である。本発明においては、ノードの数あるいはノードのレベルに制限はない。Ｂ１で表される各文書コンポーネントが別々のプリンタから供給される場合は、それらの各文書コンポーネントを別個のノードを持つ別個のジョブセグメントとし、その後ノードＢ１で互いに新しいジョブセグメントとして結合されるようにすることも可能であろう。この場合、ノードＢ１から移送された時、ジョブセグメント・スタックは分解され、そのコンポーネントは新しいジョブセグメントＰ１として並べ直されていることに注意する必要がある。多くの折丁、別丁、表紙等を備える複雑な書籍の場合は、ＶＦＪＴＤＢで多くのノードと多くのノードレベルが表されるということは明らかであろう。

【００６５】次に、図１４には、図５にブロック７００で示されるような仕上げモジュール・コーディネータ（ＦＭＣ）がより詳細に図解されている。図１４のステップ６００から始まって、各ジョブセグメント毎にＪＳＩＣを入力する２つの方法が示されている。ステップ６００Ａでは、ＶＦＪＴＲがジョブのジョブセグメントからＪＳＩＳあるいは他のＪＳＩを読み取る。上に述べたように、ＶＦＪＴＲは、バーコードのＪＳＩＣ、グリフコード、光学文字をＯＣＲで読み取り、磁気メモリまたは光メモリを読み取り、あるいは機械が解釈することができる何らかの他のコード化情報あるいは記号化情報を読み取ることが可能である。ステップ６００Ａに対して、ステップ６００Ｂには、ＪＳＩＣはオペレータ読み、入力することもできるということが示されている。ステップ６０１で、ＦＭＣはＶＦＪＴＲまたはオペレータからＪＳＩＣのデータを受け取る。ステップ６０２で、ＦＭＣは受け取ったデータがそのジョブ用の全てのジョブモデル・データを備えるかどうかを問い合わせる。上に述べたように、ジョブ関連データの完全セットをグリフまたは他の方法によってジョブの各ＪＳＩＳ上にコード化して入れることもできる。図７に示すＶＦＪＴの部分６０２にはこのような情報が入っている。ＦＭＣがオペレータまたはＶＦＪＴＲから完全なジョブモデルを受け取らなかった場合、ＦＭＣは、ステップ６０３で、関連するジョブモデル・データの完全セットをＶＦＪＴＤＢに要求する。これを行うために、ＦＭＣは、ＶＦＪＴＤＢ用のコントローラ（これはＰＭＣでもＦＭＣでも、あるいは他のコントローラでもよい）に関連するＪＳＩＣを伝えることによって、ＶＦＪＴＤＢが特定されたジョブセグメント・ノードを探すことができるようにする。ＶＦＪＴＤＢとの関連で上に説明したように、このノード内に記録されたデータを用いて、ジョブの全てのノードからの完全なジョブモデルと情報をＶＦＪＴＤＢから取り出すことができる。ステップ６０３が終わると、処理はステップ６０２に戻り、このステップでＦＭＣはジョブモデルデータの完全セットを受け取ったかどうかを再度問い合わせる。この問合わせの答えがイエスならば、ステップ６０４でＦＭＣはジョブモデル・データを読んで、ジョブの全てのジョブセグメントを特定する。同じシートを備えるジョブセグメントであっても唯一のＪＳＩＣを持つので、全てのＪＳＩＣの完全なリストはＦＭＣに完全なジョブの流れ、すなわち、単一の完成品をどのように組み上げる（ｂｕｉｌｄ）かという方法の記述だけではなく、それらの全ての製品を組み上げるために集成機／仕上げ機プロセスを通して「流す」全てのジョブセグメントのリストも供給する。また、個々の製品単位と全てのサブ単位の製造プロセス全体の両方を追跡するために利用することができるデータのインディケータを示す図１２も参照するべきである。ステップ６０５で、ＦＭＣは、ＶＦＪＴＤＢあるいは他のデータベースから、ジョブに対して特定された全てのジョブセグメントの場所及び状態に関するデータを抽出する。連続印刷作業では、ジョブセグメントの一部はやはりプリンタ内で処理待ちする場合があり、この状態がＦＭＣに伝えられる。ステップ６０６で、ＦＭＣは、ジョブモデルデータに基づいて、全てあるいは十分な量のジョブセグメントが集成機／仕上げ機の作業が実行可能でこれらの作業に使用可能であるかどうかを判断する。この判断の結果がノーならば、ステップ６０７でＦＭＣはそのことをオペレータに知らせ、次の命令を待つ。判断結果がイエスの場合は、ＦＭＣは、既に作用対象物が集成及び仕上げ作業が実行可能であると判断している。

【００６６】ＦＭＣは、ステップ６０８で装置の作業実行可能性についての調査を開始する。ステップ６０８で、ＦＭＣは、ジョブモデルから次の仕上げ作業のセットに必要な個々の装置を特定する。ほとんどの場合、集成／仕上げ作業は、連続型の集成機／仕上げ機プロセスで完成品を生産するよう設計される。しかしながら、集成機／仕上げ機プロセスがいくつかの不連続フェーズに

分解する方が好都合な場合は、ＦＭＣは、次のフェーズの作業に必要な集成機／仕上げ機装置を特定しさえすればよい。また、ステップ６０８の一部として、ＦＭＣは、ジョブが実行されるためにジョブモデルが必要とする全ての装置構成属性をジョブモデルから抽出する。例えば、集成機装置が種々異なる入力ビン配置を許容し、ジョブモデルが特定の配置を指定している場合、ジョブモデルからこのデータが抽出されることになる。同様に、紙パスガイド、折り目付け／折り装置等を自動プログラミングによって調整することができない指定された場所に置く必要がある場合、そのような属性はジョブモデルを通して特定される。ステップ６０９で、ＦＭＣは、リストされた装置にそれらの装置及び指定された構成がジョブの処理のために利用可能かどうかを問い合わせ、判断する。ＦＭＣと集成機／仕上げ機装置との間のインタフェースは種々の形を取ることができる。そのような１つのインタフェース・プロトコルとしては、ゼロックス・コーポレーション社のモジュール供給・仕上げアーキテクチャ（Ｍｏｄｕｌａｒ　Ｆｅｅｄｉｎｇ　ａｎｄ　Ｆｉｎｉｓｈｉｎｇ　Ａｒｃｈｉｔｅｃｔｕｒｅ）（ＭＦＦＡ）がある。ＭＦＦＡについては、ウェブスターに対して発行された米国特許第５，７０１，５５７号を含む多くの米国特許に記載されている。装置及び構成がその時現在利用可能でない場合、ＦＭＣは、ステップ６１０でＭＦＦＡまたは他のインタフェース・プロトコルと装置のプログラム可能性とを結びつけることによって、ＦＭＣが該当する装置を各装置及びその指定された構成を利用可能にするようプログラムすることができるかどうか問い合わせる。その答えがイエスならば、ＦＭＣはステップ６０９に戻る。答えがノーならば、ステップ６１１で、ユーザにそのことが知らされる。ユーザが、装置を所望の状態にするすることができ、かつそのように選択すると、ステップ６１２でユーザがそれらの変更を実行することの指示が出される。変更が実行されたならば、ユーザは変更が行われたことを知らせ、プロセスはステップ６０９に戻る。ユーザが必要な変更を行うことができないか、あるいはそのような変更を行わないことに決めた場合は、ユーザが抑制された条件が避けられる異なるスレッドを使用して異なるジョブモデルを生成するよう決めない限り、ジョブは一時休止される。これを行うためには、ユーザは、ステップ６１３で、図１０に示すステップ１０５にプロセスを戻し、そこで新しいスレッドが生成される。また、一組の異なる制約に遭遇する場合もあるので、新しいジョブセグメントにする方よいかもしれない。新しいスレッド作成の一部として、利用不可能な装置または構成は利用可能な装置のＶＦＪＴＤＢリストから取り除かれることになろう。ここで留意しなければならないのは、図１５にはこのタスクのためにプロセスがＰＭＣに戻されるように示されているが、このタスクはＦＭＣ内部で行うようにすることも可能であるということである。図５に示すように、ＰＭＣあるいはＦＭＣのどちらがこのタスクを実行するにしても、その結果はＶＦＪＴＤＢのジョブモデル部分に伝えられて記憶され、他のコントローラで利用することが可能である。

【００６７】ステップ６０９で、始めに選択されたジョブモデルあるいはその後手直しされたジョブモデルのために必要な装置及び構成が利用可能であることが確認されると、ＦＭＣは、ステップ６１４で、使用するよう選択された各装置に種々のジョブセグメントをロードする場所及び方法を個々に定義するタスクに進む。この情報のうちの一部は、図８乃至１１との関連で説明したＰＭＣプロセスによって既に判断が下されている場合もある。しかしながら、ビンの選択や具体的な配向（上向きか下向きか；エッジを最初に綴じるか最後に綴じるか等）のような詳細は、ＦＭＣが集成機／仕上げ機の作業のための特定のスレッド及び構成を実施するよう求められるまで最大の柔軟性が維持されるように、ジョブのフェーズ１で選択しないでおかれる場合もある。ステップ６１５で、ＦＭＣは、ロードされるべき全てのビンが実際に正しくロードされているかどうかを集成機／仕上げ機装置との情報交換を通じて確認する。必要ならば、再度命令が出される。全てのビンが正しくロードされていれば、ＦＭＣは、ステップ６１６で、ジョブモデルに従って動作するよう自動的にプログラム可能な装置を自動的にプログラムする。プログラミング命令には、それぞれの場合に応じて、ＶＦＪＴＤＢから抽出された機能データ及び制約データに基づきＦＭＣ自体によって計算されたパラメータが含まれる場合もある。例えば、ＦＭＣは、第１の装置と共に動作するもう一つの装置の制約に基づいて装置の送り速度を決定することが可能である。ＦＭＣにより自動的にプログラムすることが不可能な装置については、ＦＭＣはオペレータにそのようなプログラミングに関して指示を与える。そのための命令セットは、例えば印刷されたＪＳＩＳの形であってもよい。ステップ６１７でプログラミングが適切であると確認されたならば、ＦＭＣ機能のセットアップ部分は終了である。

【００６８】ステップ６１８を起点として、ＦＭＣの制御、追跡及び完全性確認機能が開始される。ステップ６１８で、ＦＭＣは実行コマンド（Ｒｕｎ　Ｃｏｍｍａｎｄ）を出して集成機／仕上げ機のプロセスを開始させる。あるいは、ＦＭＣは、集成機／仕上げ機装置がレディ（動作可能）状態にあることをオペレータに知らせる実行リリース（Ｒｕｎ　Ｒｅｌｅａｓｅ）信号をオペレータに対して出す。動作が開始された後、ＦＭＣは、ステップ６１９でジョブの実行を監視、追跡すると共に、ステップ６２０で追跡及び実行データに応答して適切な制御コマンドを出す。追跡機能の部分では、種々の装置のシート計数機能を用いてシートが計数される。同様

に、ＦＭＣは、各ジョブセグメントを、それらがプロセスを通って移動するのに伴って、各々のＪＳＩによって追跡することも可能である。これを行う際には、ＦＭＣは、順次装置にロードされるジョブセグメントを求める要求を出すことができ、例えば、「空ビン」信号が発生する前であってもビンに補充する命令を出すことができ、あるいは完成製品やジョブセグメントでいっぱいになったビンを空ける命令を出すことができる。また、ＦＭＣは、各装置の動作情報にアクセスすることができ、そして連続送り数を最適範囲内に保つように送り速度を調節するコマンドを出すことも可能である。さらに、ＦＭＣは、製本材料のようなサプライ品の消費具合を監視し、状況に応じて、そのようなサプライ品を補充する要求を出すこともできる。

【００６９】完全性確認及び制御機能はステップ６２１～６２９で示されている。ステップ６２１で、集成機／仕上げ機は、ＦＭＣに紙詰まりまたはジョブ停止通知を送るようプログラムされる。すると、ＦＭＣは、紙詰まりを起こした装置または停止した装置に応じて他の装置を適切に一時停止させるよう調整する。ステップ６２２で、ＦＭＣは、停止あるいは紙詰まりを起こした装置にエラー分析のための問い合わせを行う。ステップ６２３で、ＦＭＣはオペレータに指示を出す。これらの指示は、どの装置が故障の原因かをオペレータに知らせるだけの簡単なもの場合もあれば、完全な復旧指示を与えるような複雑なものになる場合もある。ステップ６２４で、ＦＭＣは紙詰まりあるいは停止状態が解消されたかどうかを確認する。この状態が解消されていなければ、シーケンスはステップ６２２に戻る。この状態が解消された場合は、ステップ６２５で、ＦＭＣは再始動コマンドを出す。再始動コマンドはオペレータ及び／または装置に対する再ロード及び再構成コマンドが含むことも可能である。再始動されたならば、ＦＭＣはステップ６２１に戻り、プロセスが続けられる。

【００７０】ステップ６２６で、ＦＭＣは完全性データを保存する。ステップ６１９及び６２０で得られた計数データ及びジョブセグメント状態データに加え、ステップ６２１～６２５で得られた停止及び紙詰まり復旧データに基づいて、ＦＭＣは、集成機／仕上げ機プロセスの間にどのシートが失われ、破損したか、あるいは行方不明であるかを追跡する。また、ＦＭＣは、計数データ及び追跡のデータに基づいてどのジョブセグメントまたは完成品に落丁がある可能性があるかを追跡する。この完全性データに基づいて、ＦＭＣは、ステップ６２７で、追加印刷及び／または集成機／仕上げ機の作業のための指示を出す。これらの指示は自動的に該当する装置に出すことも可能であり、あるいはジョブのある部分のマニュアル再プログラミングに関するオペレータへの指示の場合もある。集成機／仕上げ機動作の全ての部分を通して複雑なジョブでも追跡することができるＦＭＣを持つことのＩつの利点は、各文書コンポーネント毎に部数を余分に印刷する動機が弱くなり、そのことによって印刷コスト及び在庫コストが節減されるということである。このような動機が弱くなるのは、ＦＭＣがジョブセグメント及びジョブセグメント中のシートを追跡する機能を有することによって、オペレータがいつ、どこで欠陥が生じたかをより正確に知ることができるからである。

【００７１】ステップ６２８で、ＦＭＣは、ジョブ実行の記録が取られるように、その追跡データ及び完全性データをＶＦＪＴＤＢのような中央データベースへ送る。このステップ６２８は、生産工程の終了時に行われるが、任意の中間段階で行うこともできる。ＰＭＣあるいは他のコントローラは、このデータを用いて、完成ユニットの数、廃棄シートの量、生産に要した時間、及び監視し記録することが望ましいその他の基準条件を把握することができる。最後にステップ６２９で、ＦＭＣは、装置に関連する何らかの新しい機能及び制約データに関するＶＦＪＴＤＢを更新する。このようなデータとしては、例えば信頼性データ、運転休止データ、送り上の制約に関する新しい情報等が含まれる。要するに、集成機／仕上げ機ジョブを追跡することが可能なジョブセグメントに分解することによって、ＦＭＣは、今まで互いにあるいはプリンタのような装置に最小限にしかディジタル接続されなかったオフライン集成機／仕上げ機装置の体系的制御及び完全性監視を可能にする。

【００７２】次に、本発明のためのソフトウェアの重要な部分の実施例について説明する。ＶＦＪＴＤＢに関して上に説明したように、各ジョブは、不定の数の文書コンポーネント及び／またはジョブセグメントで構成され得る。このように文書コンポーネントやジョブセグメントの数が不定という特徴は、スタティックメモリ・バッファのような定義済みの制限や限度を使用しないソフトウェア・アルゴリズムが必要になる。その代わりに、ソフトウェアアルゴリズムは、ジョブのキー順に基づいてＶＦＪＴＤＢあるいは他のデータベース中のジョブモデルを再帰的に探索する。ジョブのキー順は、個々のジョブセグメントと関連付けられたデータベース・レコードから得られる。マイクロソフト・ビジュアル・ベーシック（Ｖｉｓｕａｌ　Ｂａｓｉｃ）６．０の定義済みトリービュー・コンポーネント（Ｐｒｅｄｅｆｉｎｅｄ　ＴｒｅｅＶｉｅｗ　Ｃｏｍｐｏｎｅｎｔ）を用いることにより、「ノードキー（Ｎｏｄｅ　Ｋｅｙ）」がジョブキー順を用いて構築され記憶される。これによって、ユーザはグラフィカル・ディスプレイ上でノードをアイコンの形で選択し、ジョブシーケンスキーは「ノードキー」の形で検索することが可能になる。すると、データベース・ジョブモデルは、処理要件が容易化されるように再帰的に探索することができる。

【００７３】下記のソフトウェアは、構築中の物理的製品のグラフィカルＴｒｅｅＶｉｅｗ表現を構築するため

41

に使用される再帰的アルゴリズムの実施態様を示したものである。ノードキーは親ノードキーのシーケンスとして組み立てられる。ノードアイコンが選択された後の任意の時点で、ノードキーにアクセスし、それを用いてジョブモデル全体にわたって効率的にノードを探し出すことができる。このサブルーチンは、それ自身を再帰的にコールすることによってトリーをそのルートから構築し、それ以上子が存在しない時、そのトリーの各分岐で停止する。このアルゴリズムを使用することには次のような利点がある：

１．ジョブモデル・トリーに存在するレベル数について＊

```
Private Sub FillTree(Childid As String, Key As String,_
          nodeindex As Integer, nodx As Node, keys() As String,_
          level As Integer, nf As Form)
'a recursive building of the job node tree （ジョブノードトリーの再帰的構築）
Dim pd As NodeDef
Dim pdkey As String
NumChildren = GetChildren(Childid, pd)
If (NumChildren > 0) Then
  For x = 1 To NumChildren
  pdkey = Key & ",," & pd.Nodelds(x) & ",," & Str$(nodeindex)
  nodeindex = nodeindex + 1
  Set nodx = nf.TreeViewl .Nodes.Add(Key, tvwChild, pdkey,_
                pd.Nodelds(x) & " "& pd.Dscr(x))
  'Seems senseless but this makes entire Tree Visible （無意味に見えるがこれによって全体的にトリービジブル（Ｔｒｅｅ　Ｖｉｓｉｂｌｅ）になる）
  nodx. Selected = True
  nodx. Selected = False
  If (keys(level) = pd.Nodelds(x)) Then nodx.Checked = True
  Call FillTree(pd.Childlds(x), pdkey, nodeindex, nodx, keys, level + 1,
 nf)
 Next x
End If
End Sub
```

【００７５】本発明のＦＭＣの部分のもう一つのアルゴリズムでは、それ自身を再帰的にコールしてトリーの全てのノードが所要の状態であるかどうかをチェックするサブルーチンが得られる。その状態にないノードが１つ見つかるまで、ジョブモデル・トリー全体にわたる全てのノードがチェックされる。この場合、仕上げ装置は各ビンがロードされた状態かどうか確かめるためのチェックなされている。関数ＣｈｅｃｋＳｅｇｍｅｎｔｓは真または偽の値をリターンする。このサブルーチンの実際の目的はＣｈｅｃｋＳｅｇｍｅｎｔｓの代わりに単に新しい関数を用いるだけで容易に変更することができる。※

42

＊定義済みの限界が全く定義されない；

２．コードサイズがコンパクトである；

３．コードは、ビルトインの既存のマイクロソフト・ビジュアル・ベーシック・コントロールズ（Ｍｉｃｒｏｓｏｆｔ　Ｖｉｓｕａｌ　Ｂａｓｉｃ　Ｃｏｎｔｒｏｌｓ）を用いてグラフィカル表示を提供する；

４．コードは、ユーザ選択イベントを処理する時、シーケンスキーを効率的に検索する。

【００７４】このソフトウェア・コードは下記の通りである：

```
Private Sub JobRun Tree(Childid As String, Abort As Boolean,_
          Formldx As Integer,_
          JobldStr As String,_
          nf As Treeview,_
```

※そのために、このコードは、複製してＦＭＣで多くの目的のために使用することが可能になる。

【００７６】このアルゴリズムを使用することには次のような利点がある：

１．ジョブモデル・トリーに存在するレベル数について定義済みの限界が全く定義されない；

２．コードサイズがコンパクトである；

３．コードを複製して多くのＦＭＣのタスク及び関連タスクのために再使用することが容易である。

【００７７】ソフトウェアの一例を以下に示す：

```
        Nodeldx As Variant)
'recursive traverse of the job node tree used buy Sub JobRun
Dim pd As NodeDef
Dim NumChildren As Integer
If (Abort = False) Then
 NumChildren = GetChildren(Childid, pd)
 If (CheckSegments(Formldx, JobldStr, NumChildren,_
      nf.Nodes(Nodeldx).Key, Nodeldx)) Then
  Abort = True
 Else
  If (NumChildren > 0) Then
   For x = 1 To NumChildren
    Call JobRunTree(pd.Childlds(x), Abort, Formldx, JobldStr, nf, Nodeld
x)
   Next x
  End If
 End If
End If
End Sub
```

【００７８】本発明のもう一つの態様は、ユーザが電子画像データファイルを特定の文書コンポーネントにリンクし、その後文書コンポーネントを文書の形に編成するのを支援するソフトウェア・ユーザインタフェース及び「文書編成ウィザード」（ＤＣＷ）ソフトウェアである。ユーザは、作りたい所望の文書を構成する画像を表す電子画像データファイルを作成する。ユーザは、リストされた文書様式の種類の１つから最終文書様式を選択する。次に、ユーザは各画像データファイルを７つのうちの１つの文書様式のうちの一特定の文書コンポーネントとして指定する。このソフトウェアは、ユーザが文書様式中における文書コンポーネントの配向及び順番を指定できるようにし、ウィザードはこのような指定を容易にする。また、このソフトウェアは、各文書様式毎に異なる一組の規則を使用して、ユーザがそれから文書を編成することができる文書コンポーネントの数と種類を規制する。例えば、折丁小冊子は、表紙である２枚の別個の媒体を持つことができない。折丁小冊子は１枚の表紙（ラップアラウンド表紙）しか持つことができないが、テープ綴じ文書様式では表紙を２つ（表紙と裏表紙）を持つことが可能である。このソフトウェアは、選択された文書コンポーネントの種類及び／または文書様式によって決まる必要属性を入力するようユーザに促す。また、このソフトウェアは、ユーザが利用することができる仕上げ装置の機能及び制約を表すファイルまたはデータベースを読み、従って利用可能なその仕上げ装置に基づいて利用することができる仕上げ作業、仕上げオプションあるいは属性のみを表示することになろう。

【００７９】ＤＣＷは、コンポーネントが可変コンポーネント（Ｖａｒｉａｂｌｅ　Ｃｏｍｐｏｎｅｎｔ）かどうかを入力するようユーザに促すことも行う。可変コンポーネントは、印刷ジョブ中の１部の文書と次の１部の文書とで画像データが変わるコンポーネントである。また、固定コンポーネントとは、印刷ジョブで１部の文書と次の１部の文書とで画像データが変わらないコンポーネントである。コンポーネントが可変コンポーネントとして指定される場合、ユーザは様式ファイル名及びデータファイル名を指定することが必要になろう。このデータは全て例えばデータベースに保存される。

【００８０】ＤＣＷは、ユーザが、いずれかの文書コンポーネントが外部コンポーネント（Ｅｘｔｅｒｎａｌ　Ｃｏｍｐｏｎｅｎｔ）であるかどうか、あるいはその文書コンポーネントは画像データファイルから生成するべきであるかどうかを指定できるようにする。外部コンポーネントとは、何らかの他のプロセスを通して生成されるか印刷されるが、このジョブの一部としては印刷されないコンポーネントである。コンポーネントが外部コンポーネントとして指定された場合、ＤＣＷは、前に印刷された文書コンポーネント用の完全性記述子を入力するようユーザに促すことになる。完全性記述子は、ユーザが完全性記述子ファイル名を入力すると、ファイルからを入力するか読み出すことができる。このデータは全て例えばデータベースに保存される。

【００８１】次に、図１７乃至１９を参照して、グラフィカル・ユーザインタフェース（ＧＵＩ）を備える本発明のもう一つの実施態様を説明する。このＧＵＩは、オペレータが個々の製品コンポーネントをどのように組み上げて最終の文書製品を形成するかについての専門知識を持っていない場合においても、オペレータが集成機／仕上げ機装置に文書コンポーネントをロードして操作するのを容易にすることができるよう構成されている。従来技術においては、多くの場合、ジョブのクリエータが

機械のオペレータを直接支援するか、装置を直接操作するか、あるいは一組の詳細なプロセス指示を書くことが必要であった。

【００８２】本発明のＧＵＩによれば、仕上げ装置によって編成される文書についての予備知識を全く持っていないオペレータが必要とする詳細な情報が提供される。このインタフェースによれば、簡単なジョブ概要から非常に詳細なコンポーネント状態情報に到る情報が提供される。情報は、製品及びそのコンポーネントの各々について記述するオペレータに与えられる。また、オペレータは、様々な仕上げ装置にコンポーネントをロードして操作する方法全般にわたって案内される。また、このＧＵＩは、オペレータがＶＦＪＴＤＢあるいは他のデータベースと対話することを可能にする。このＧＵＩは、仕上げ装置の所にいるユーザが仕上げ装置を稼働してコンポーネントをロードしかつジョブをステージすることができるように設計される。また、このユーザインタフェースを用いて、仕上げ装置あるいはプリンタの出力場所におけるパッケージング作業を容易化することも可能である。例えば、オペレータは、このＧＵＩを使用して、他の場所へ送られる箱詰めされた生産物のパレット化を支援することもできよう。また、このＧＵＩは、遠隔の仕上げ場所へ輸送されるセグメントをパッケージングするためにプリンタの出力場所で使用することができる。

【００８３】次に、図１７には、ＧＵＩの初期画面の一実施例が示されている。この画面を用いて、オペレータはデータベース及びジョブ名を選択することができる。また、この画面によってユーザは、データベース中の文書コンポーネントノードを加えたり、取り除いたり、あるいはこれらと同様の操作を行うことができる。

【００８４】次に、図１８及び１９には、本発明のジョブ集成画面（Ｊｏｂ　Ａｓｓｅｍｂｌｙ　Ｖｉｅｗ）の２つの実施例が示されている。これらの両方の実施例のジョブ集成画面も、階層木構造として構成されたジョブモデルのグラフィカル・ビューが表示される。この階層木構造の画面構成はマイクロソフト・ビジュアル・ベーシック６．０の定義済みトリービュー・コンポーネントを用いて作成することができる。ジョブ集成画面は、テキスト記述でもグラフィック・アイコンでも使用することができる。例えば、折丁本のピクチャあるいは本の表紙のピクチャを表示することもできるし、製品の実ビットマップ写真画像を表示することもできる。

【００８５】図１８は、上に説明した唯一のジョブ識別子コードからジョブを参照するジョブ集成画面の一実施例を示す。図１９は、ジョブのコンボボックス状態リストからジョブを参照するジョブ集成画面の一実施例を示す。

【００８６】本発明のＧＵＩの使用には下記のような特徴がある：第１に、ユーザはジョブセグメント識別子シート（ＪＳＩＳ）を目で走査するか、あるいは他の仕方でシステムが上に説明した手順を用いてジョブセグメントのＪＳＩを決定できるようにする。上に述べたように、ＰＭＣは、ＶＦＪＴＤＢあるいは他のデータベースと共に動作して、単一のＪＳＩからジョブモデル及びその全てのノードを再集成することができる。ジョブとその全てのノードが特定されたならば、ＰＭＣ、ＦＭＣあるいは他の端末は、ユーザが「セグメント通知（Ｓｅｇｍｅｎｔ　Ｎｏｔｉｆｉｃａｔｉｏｎ）」を選択した場合、そのジョブ用のジョブ集成画面を生成することができる。

【００８７】第２に、ユーザは標準的なコンボボックスにあるジョブの状態リストを見て、どのジョブを見たいか選ぶことができる。

【００８８】第３に、ユーザはツールボックスのドラッグ・アンド・ドロップ操作によりジョブ集成画面のグラフィカル・トリーを編集することが可能である。例えば、ユーザは、製品にコンポーネントを加え、あるいは製品からコンポーネントを取り去ることができる。

【００８９】第４に、無制限に多くのジョブ集成画面のウィンドーを表示することができる。この特徴は、ジョブを後の処理のためにステージするのに役立てられる。

【００９０】第５に、ＧＵＩの主表示形式は、プリンタ及び／または集成機／フィーダ装置の待ち行列状態であるジョブ状態を表示する。

【００９１】第６に、ＪＳＩＣのようなセグメント識別子を表示しキーボードを使用して編集することが可能である。

【００９２】第７に、ユーザがジョブが実行されるよう要求した時、システムソフトウェア（これはＰＭＣ、ＦＭＣ、またはその両方あってもよく、あるいは他のクライアントの所にあってもよい）は、ＧＵＩに各ジョブセグメントの状態チェックを表示するよう指示し、欠落したあるいは不完全なジョブセグメントがあると、そのことをユーザに知らせる。ＧＵＩのこの特徴については、上にＦＭＣとの関連で説明した。

【００９３】第８に、システム診断表示形式が主表示形式からイネーブルにされアクセスされる。

【００９４】第９に、ジョブノード、文書ノード、文書コンポーネントノード、ジョブセグメントノード等に関する情報（ただし、これらに限定されない）を含め、全てのノードついての情報の種類及びレベルあるいはジョブモデルのレベルをユーザによって選択すること可能である。

【００９５】第１０には、本発明のジョブ集成画面によって表示される製品のトリー表現は、サイズ及びレベル数共無限である。最高のレベルは、様々な文書、本及びその他の品物（ＣＤ　ＲＯＭディスクのような）で満たされたパレット積みの箱でもよく、他方同じジョブ集成画面のトリーが、折丁本のただ１ページを備える最低レベルになることもある。ＰＭＣとの関連で上に説明した

ように、個々のＰＤＬはそのようなシートと関連付けられ、ジョブ集成画面のトリーのＧＵＩから特定して取り出すことができる。

【００９６】要約すると、本発明は、広範にわたる仕上げプロセスの電子管理・制御のためのシステムで、それ自身異なるジョブ間で高度に可変な作用対象物にジョブに応じて適用することができる多数の生産工程及び装置からの入力を特徴とするシステムである。本発明は、印刷文書の印刷及び仕上げ作業との関連で実施例により説明した。しかしながら、本発明の原理は、本発明は、繊維製品の製造（捺染、裁断、縫製及び仕上げを含む）、様々な消費者向け製品及び産業用製品のパッケージング作業、プリント配線基板の製造等（ただし、これらに限定されない）のような生産及び仕上げシステムに応用可能である。特に、本発明は、作用対象物の製造プロセスがそれらの作用対象物の仕上げ及びパッケージング作業のプロセスとは別個に管理されるような多くの作業に適用可能である。

【００９７】中でも、本発明においては下記のような長所が実証されている：

１）プリプレス／印刷／仕上げプロセスの３つの全てのフェーズの対話型制御、追跡及び完全性確認機能のための統合型ディジタルアーキテクチャ；

２）ジョブの印刷前に、特にプリンタコントローラとオフラインである場合に、オペレータが複合した集成／仕上げ作業に完全な指示を与えることを可能ならしめる統合された装置及び方法；

３）プリンタコントローラ及び集成機／仕上げ機コントローラがどちらもそれぞれの印刷作業及び集成／仕上げ作業を指示し制御することができるような形で複雑な文書の最終文書様式を正確かつ完全に記述する方法；

４）利用可能な印刷システム及び利用可能な仕上げシステムの両方の制約に応じて印刷ジョブ及び関連するワークフローを分割し管理する装置及び方法；

５）指定されたプリンタ及び指定された集成機／仕上げ機装置の可用性に従って印刷・仕上げプロセス全体を管理する待ち行列管理システム；

６）各製造プロセスを通して、特にプリンタコントローラとオフラインの集成機／仕上げ機の作業を含むプロセス部分を通して各シートを追跡することができる対話型の完全性確認システム；

７）対話型の完全性確認機能に応じて機能し、集成機／仕上げ機装置がシートを要求してそれらのシートが得られない場合にプリンタにそれらのシートを刷り増すかあるいは交換するように自動的に指示ないしは命令する装置及び方法。

【００９８】従って、本発明は、上に述べた目的及び長所を完全に達成するということは明白である。本発明は、いくつかの実施例との関連で説明したが、当業者には多くの代替態様、修正態様及び変形態様が自明であることは明白である。従って、従って、そのような代替態様、修正態様、及び変形態様は全て特許請求の範囲の記載に基づく発明の精神及び広義の範囲に包含されるものとする。

【図面の簡単な説明】

【図１】コンポーネント及び複雑さのレベルが種々異なる文書の仕上げ作業の例を示す説明図である。

【図２】コンポーネント及び複雑さのレベルが種々異なる文書の仕上げ作業の例を示す説明図である。

【図３】適度な複雑さの文書用の典型的な集成機／仕上げ機装置の動作を示す説明図である。

【図４】ＣＩＰ３を使用して記述することができる情報の形態を概略形式で示す説明図である。

【図５】本発明を使用した仕事の流れを示すと共に、本発明を使用する様々な装置間の関係を図解したブロック図である。

【図６】ＰＭＣと仮想仕上げジョブチケット・データベース（ＶＦＪＴＤＢ）との関係を含めて、ＰＭＣの典型的な入力及び出力を示すブロック図である。

【図７】ジョブセグメント識別子シート（ＪＳＩＳ）の一例を示す説明図である。

【図８】本発明のＰＭＣの論理型アーキテクチャの一実施例の概要を図解したブロック図である。

【図９】概ね図８のステップ７０～７３に対応するＰＭＣの部分の一実施例を示すフローチャートである。

【図１０】概ね図８のステップ７４～７６に対応するＰＭＣの部分の一実施例を示すフローチャートである。

【図１１】概ね図８のステップ７７及び７８ａ～７８ｃに対応するＰＭＣの部分の一実施例を示すフローチャートである。

【図１２】種々のジョブセグメントが通る様々なスレッドの例を図解した説明図である。

【図１３】折丁文書の生成に適用されるようなトリー構造の例を示す説明図である。

【図１４】仕上げモジュール・コーディネータ（ＦＭＣ）の動作をより詳細に図解したフローチャートである。

【図１５】仕上げモジュール・コーディネータ（ＦＭＣ）の動作をより詳細に図解したフローチャートである。

【図１６】仕上げモジュール・コーディネータ（ＦＭＣ）の動作をより詳細に図解したフローチャートである。

【図１７】ＧＵＩの初期画面の一実施例を示す説明図である。

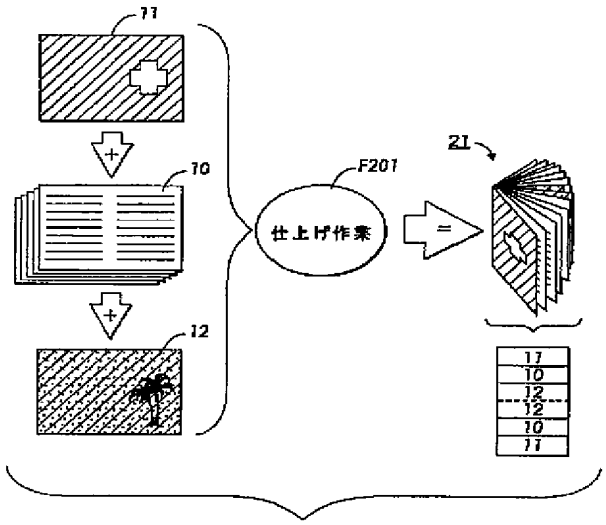
【図１８】唯一のジョブ識別子コードからジョブを参照するジョブ集成画面の一実施例を示す説明図である。

【図１９】ジョブのコンボボックス状態リストからジョブを参照するジョブ集成画面の一実施例を示す説明図である。

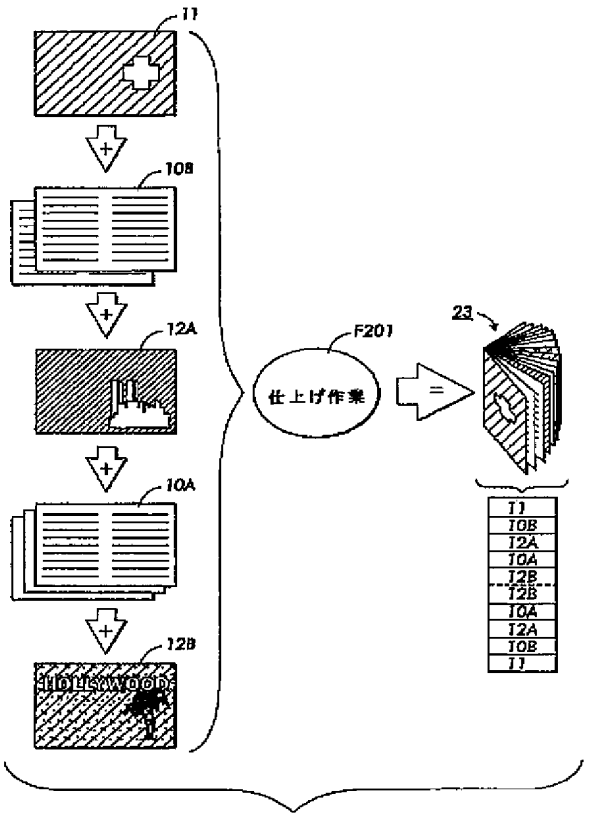
【図２０】文書属性、文書様式セレクタ、文書コンポーネント・セレクタ、ジョブ中のコンポーネントを示す図＊

＊である。

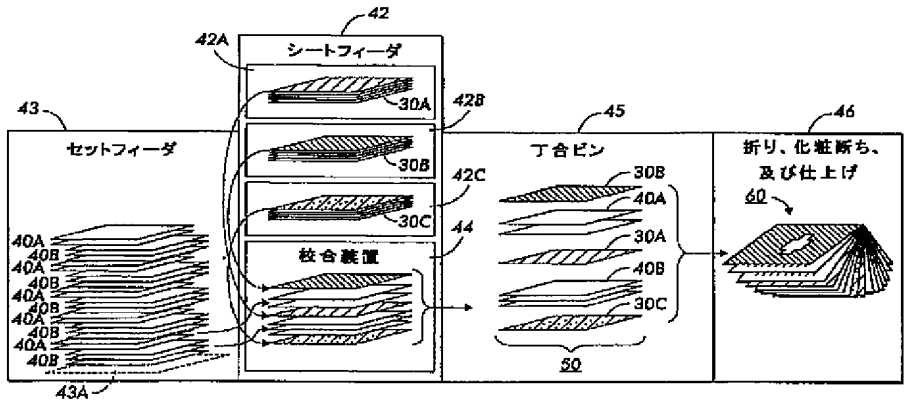
【図１】



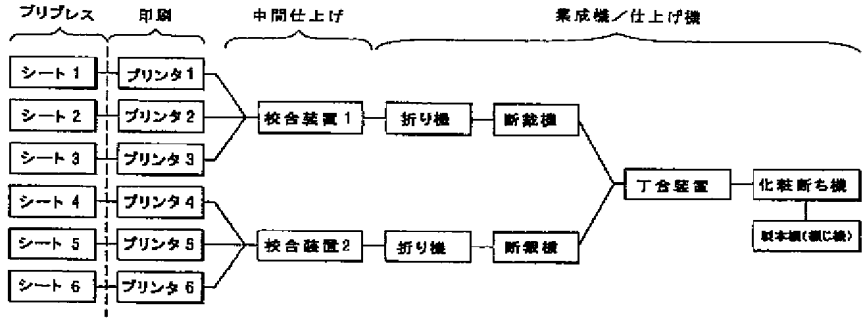
【図２】



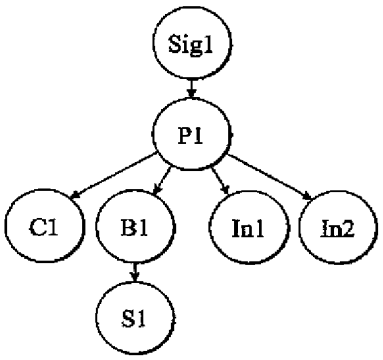
【図３】



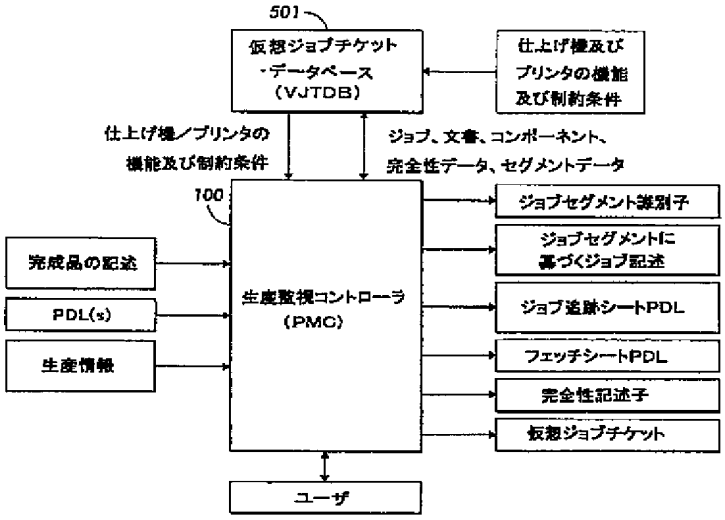
【図４】


プリプレス
印刷
中間仕上げ
集成機／仕上げ機
シート１
シート２
シート３
シート４
シート５
シート６
プリンタ１
プリンタ２
プリンタ３
プリンタ４
プリンタ５
プリンタ６
校合装置１
校合装置２
折り機
折り機
断裁機
断裁機
丁合装置
化粧断ち機
製本機(糸とじ機)

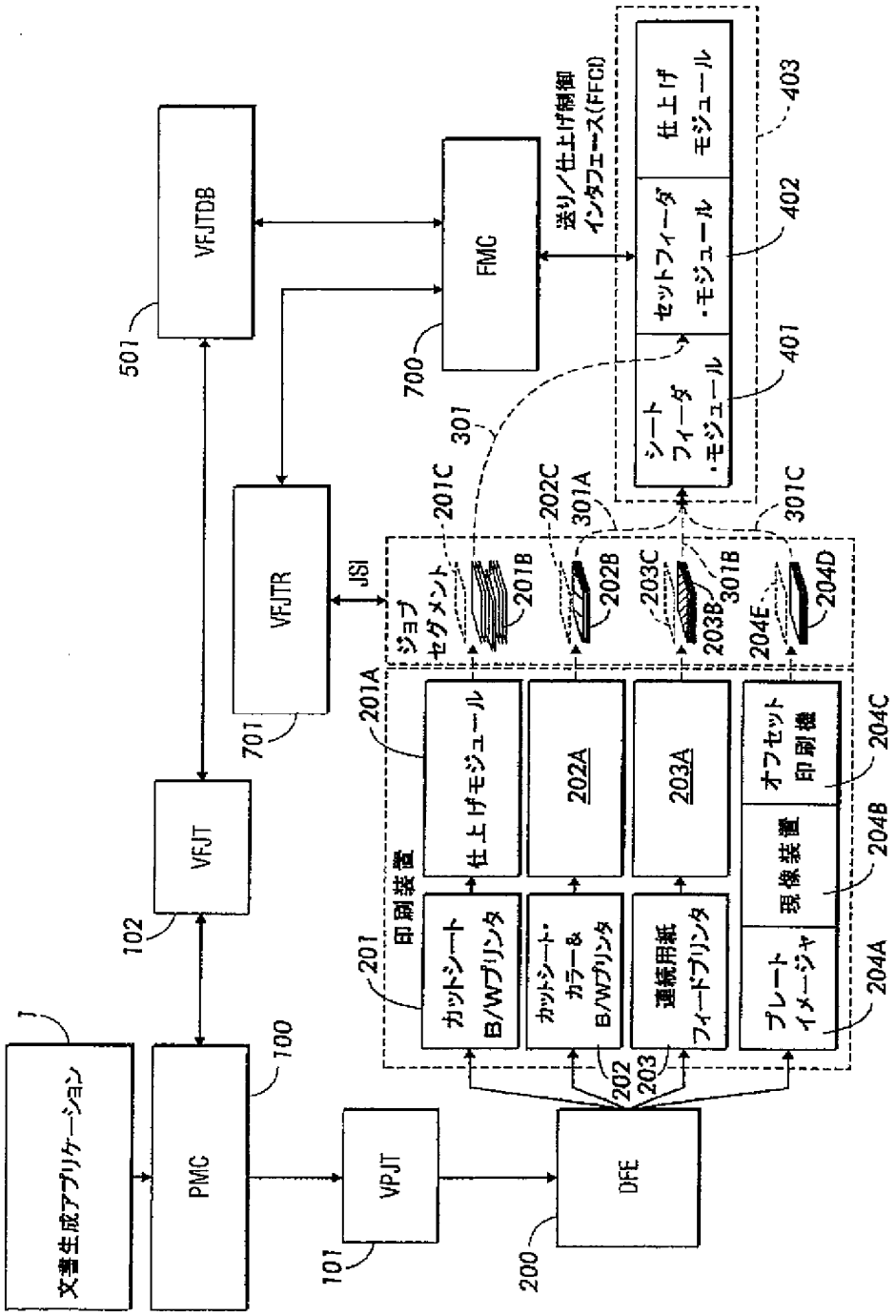
【図１３】


Sig1
P1
C1
B1
In1
In2
S1

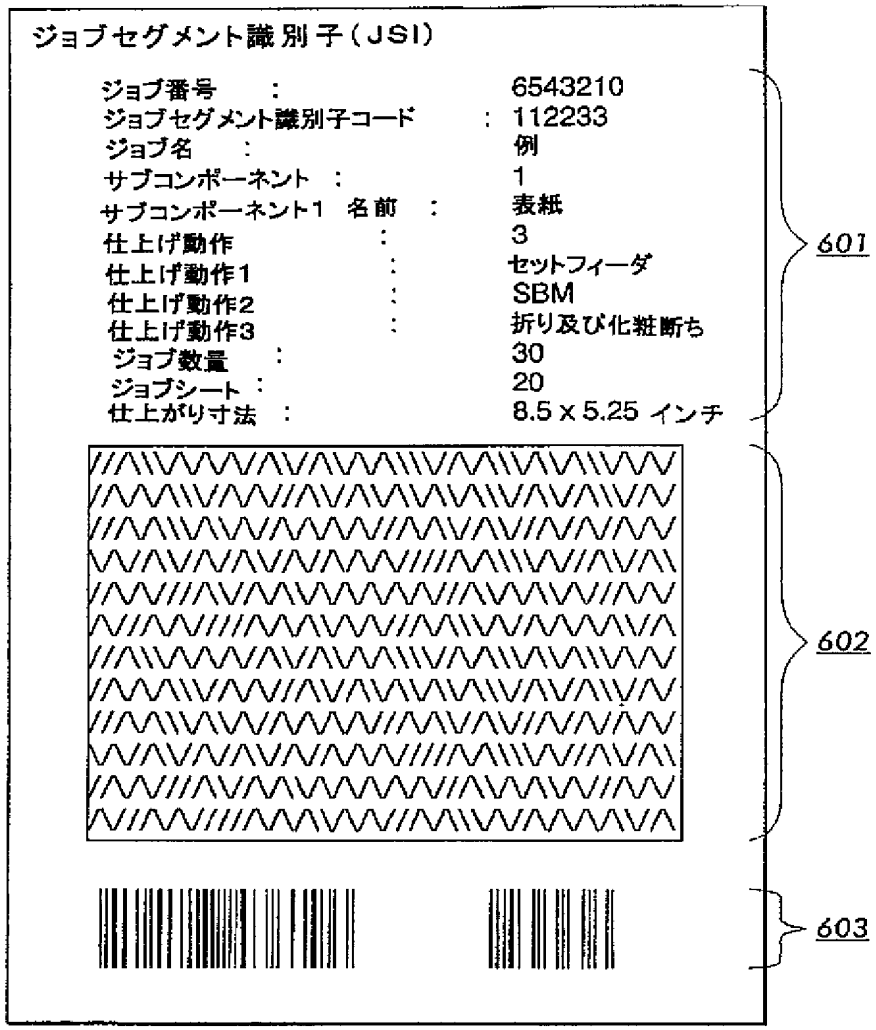
【図６】


501
仮想ジョブチケット
・データベース
（VJTDB）
仕上げ機及び
プリンタの機能
及び制約条件
仕上げ機／プリンタの
機能及び制約条件
ジョブ、文書、コンポーネント、
完全性データ、セグメントデータ
100
完成品の記述
PDL(s)
生産情報
生産監視コントローラ
（PMC）
ジョブセグメント識別子
ジョブセグメントに
基づくジョブ記述
ジョブ追跡シートPDL
フェッチシートPDL
完全性記述子
仮想ジョブチケット
ユーザ

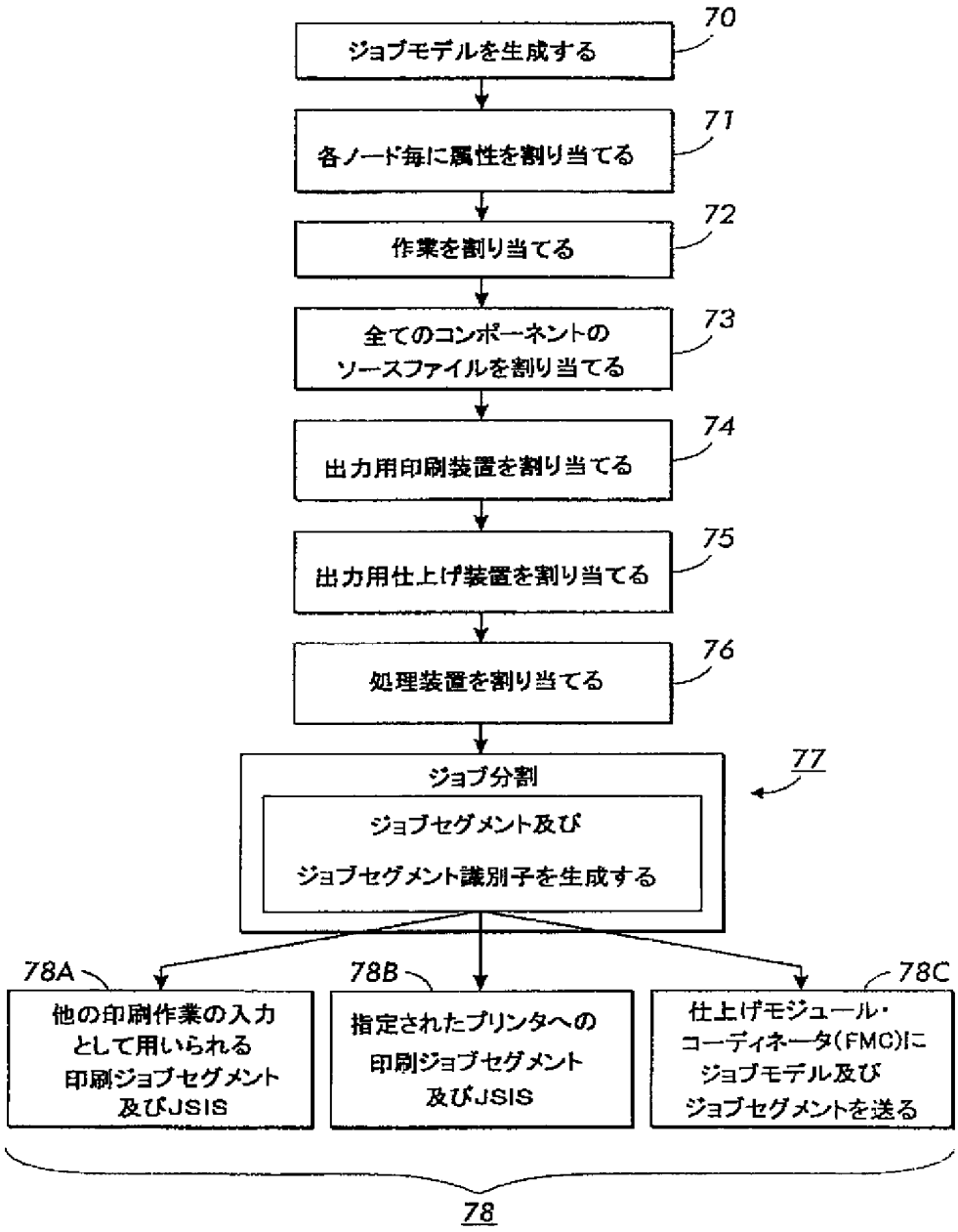
【図５】


文書生成アプリケーション
1
PMC
100
VPJT
101
DFE
200
VFJT
102
VFJTDB
501
VFJTR
701
JSI
FMC
700
送り/仕上げ制御
インタフェース(FFCI)
403
仕上げ
モジュール
402
セットフィーダ
・モジュール
401
シート
フィーダ
・モジュール
301
印刷装置
201
ジョブ
セグメント
201A
仕上げモジュール
202A
203A
201B
202B
203B
204B
201C
202C
203C
301A
301B
301C
204D
204E
カットシート
B/Wプリンタ
カットシート・
カラー&
B/Wプリンタ
連続用紙
フィードプリンタ
プレート
イメージャ
204A
現像装置
オフセット
印刷機
204C
202
203

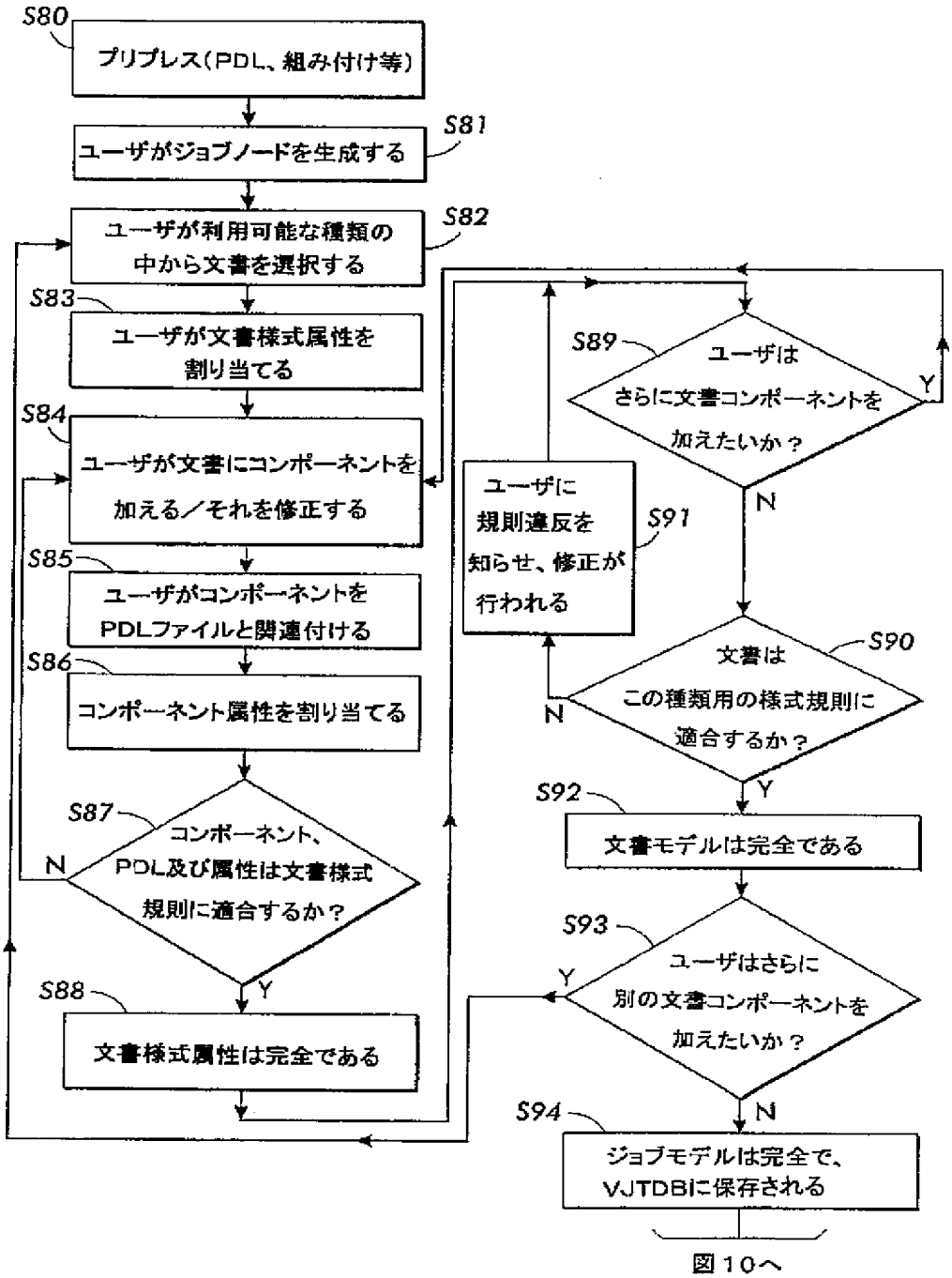
【図７】


ジョブセグメント識別子（JSI）
ジョブ番号 : 6543210
ジョブセグメント識別子コード : 112233
ジョブ名 : 例
サブコンポーネント : 1
サブコンポーネント1 名前 : 表紙
仕上げ動作 : 3
仕上げ動作1 : セットフィーダ
仕上げ動作2 : SBM
仕上げ動作3 : 折り及び化粧断ち
ジョブ数量 : 30
ジョブシート : 20
仕上がり寸法 : 8.5 x 5.25 インチ
601
602
603

【図８】


70
ジョブモデルを生成する
71
各ノード毎に属性を割り当てる
72
作業を割り当てる
73
全てのコンポーネントの
ソースファイルを割り当てる
74
出力用印刷装置を割り当てる
75
出力用仕上げ装置を割り当てる
76
処理装置を割り当てる
77
ジョブ分割
ジョブセグメント及び
ジョブセグメント識別子を生成する
78A
他の印刷作業の入力
として用いられる
印刷ジョブセグメント
及びJSIS
78B
指定されたプリンタへの
印刷ジョブセグメント
及びJSIS
78C
仕上げモジュール・
コーディネータ(FMC)に
ジョブモデル及び
ジョブセグメントを送る
78

【図９】


S80
プリプレス(PDL、組み付け等)
S81
ユーザがジョブノードを生成する
S82
ユーザが利用可能な種類の中から文書を選択する
S83
ユーザが文書様式属性を割り当てる
S84
ユーザが文書にコンポーネントを加える／それを修正する
S85
ユーザがコンポーネントをPDLファイルと関連付ける
S86
コンポーネント属性を割り当てる
S87
コンポーネント、PDL及び属性は文書様式規則に適合するか？
N
Y
S88
文書様式属性は完全である
S89
ユーザはさらに文書コンポーネントを加えたいか？
Y
N
S90
文書はこの種類用の様式規則に適合するか？
N
Y
S91
ユーザに規則違反を知らせ、修正が行われる
S92
文書モデルは完全である
S93
ユーザはさらに別の文書コンポーネントを加えたいか？
Y
N
S94
ジョブモデルは完全で、VJTDBに保存される
図１０へ

【図１０】

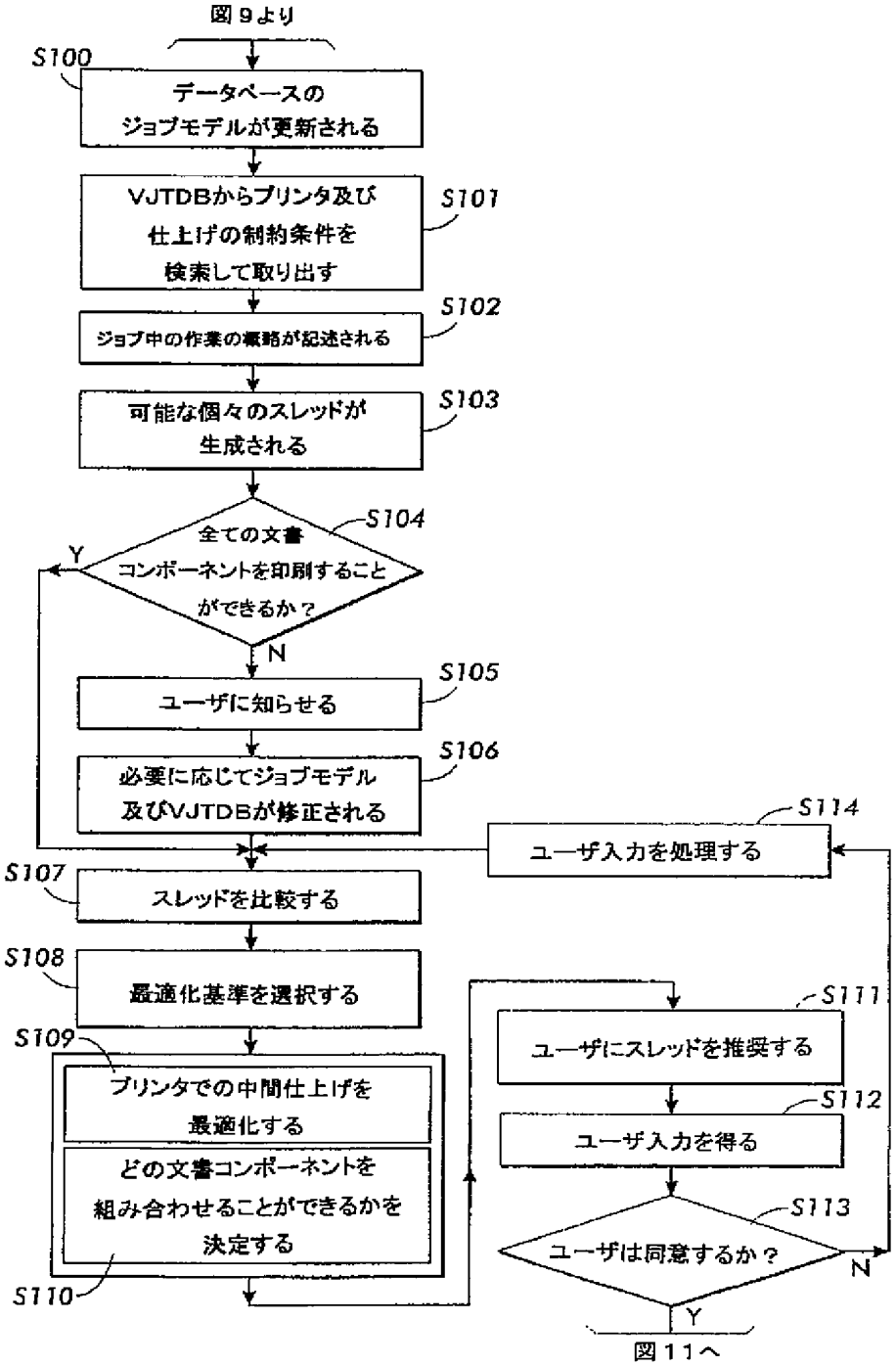


図９より
S100
データベースの
ジョブモデルが更新される
VJTDBからプリンタ及び
仕上げの制約条件を
検索して取り出す
S101
ジョブ中の作業の概略が記述される
S102
可能な個々のスレッドが
生成される
S103
全ての文書
コンポーネントを印刷すること
ができるか？
S104
Y
N
ユーザに知らせる
S105
必要に応じてジョブモデル
及びVJTDBが修正される
S106
ユーザ入力を処理する
S114
スレッドを比較する
S107
最適化基準を選択する
S108
プリンタでの中間仕上げを
最適化する
どの文書コンポーネントを
組み合わせることができるかを
決定する
S109
S110
ユーザにスレッドを推奨する
S111
ユーザ入力を得る
S112
ユーザは同意するか？
S113
N
Y
図１１へ

【図１１】

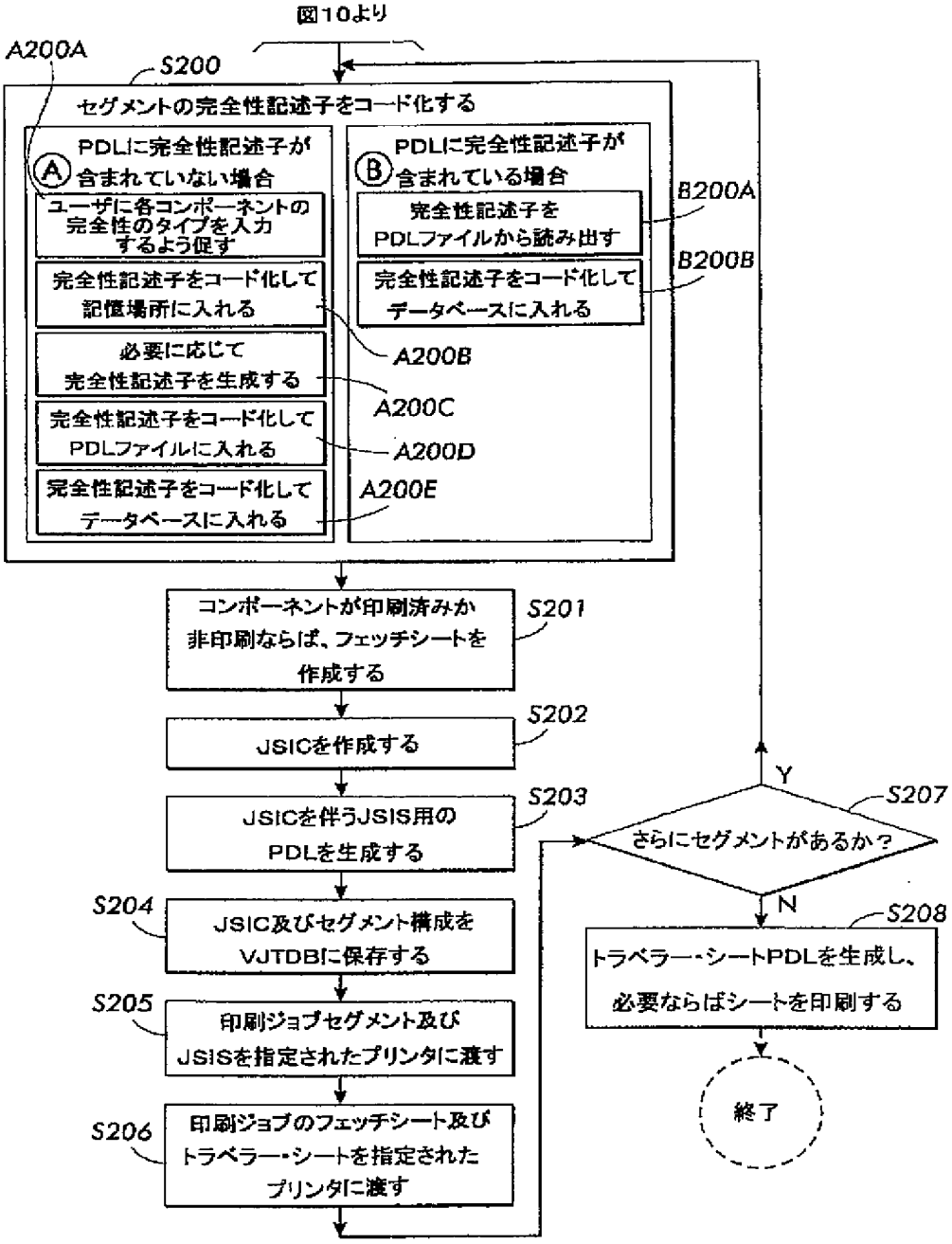
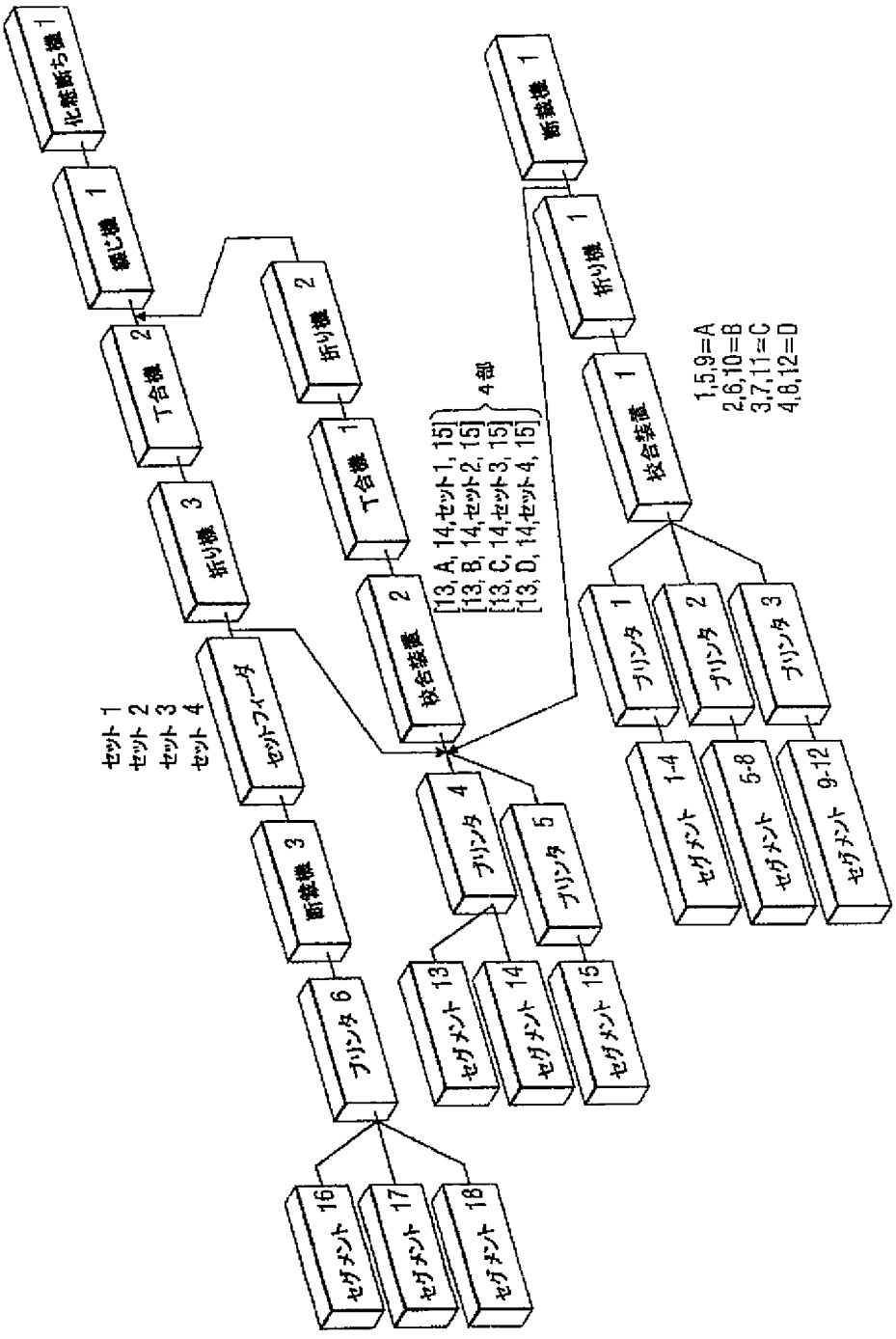


図10より
A200A
S200
セグメントの完全性記述子をコード化する
PDLに完全性記述子が含まれていない場合
Ⓐ
ユーザに各コンポーネントの完全性のタイプを入力するよう促す
完全性記述子をコード化して記憶場所に入れる
必要に応じて完全性記述子を生成する
完全性記述子をコード化してPDLファイルに入れる
完全性記述子をコード化してデータベースに入れる
PDLに完全性記述子が含まれている場合
Ⓑ
完全性記述子をPDLファイルから読み出す
完全性記述子をコード化してデータベースに入れる
B200A
B200B
A200B
A200C
A200D
A200E
コンポーネントが印刷済みか非印刷ならば、フェッチシートを作成する
S201
JSICを作成する
S202
JSICを伴うJSIS用のPDLを生成する
S203
JSIC及びセグメント構成をVJTDBに保存する
S204
印刷ジョブセグメント及びJSISを指定されたプリンタに渡す
S205
印刷ジョブのフェッチシート及びトラベラー・シートを指定されたプリンタに渡す
S206
さらにセグメントがあるか？
S207
Y
N
トラベラー・シートPDLを生成し、必要ならばシートを印刷する
S208
終了

【図１２】


化繊断ち機 1
綴じ機 1
T合機 2
折り機 3
セットフィーダ
断裁機 3
プリンタ 6
セグメント 16
セグメント 17
セグメント 18
セット 1
セット 2
セット 3
セット 4
折り機 2
T合機 1
校合装置 2
[13, A, 14,セット1, 15]
[13, B, 14,セット2, 15]
[13, C, 14,セット3, 15]
[13, D, 14,セット4, 15]
4部
プリンタ 4
プリンタ 5
セグメント 13
セグメント 14
セグメント 15
断裁機 1
折り機 1
校合装置 1
1,5,9=A
2,6,10=B
3,7,11=C
4,8,12=D
プリンタ 1
プリンタ 2
プリンタ 3
セグメント 1-4
セグメント 5-8
セグメント 9-12

【図１４】

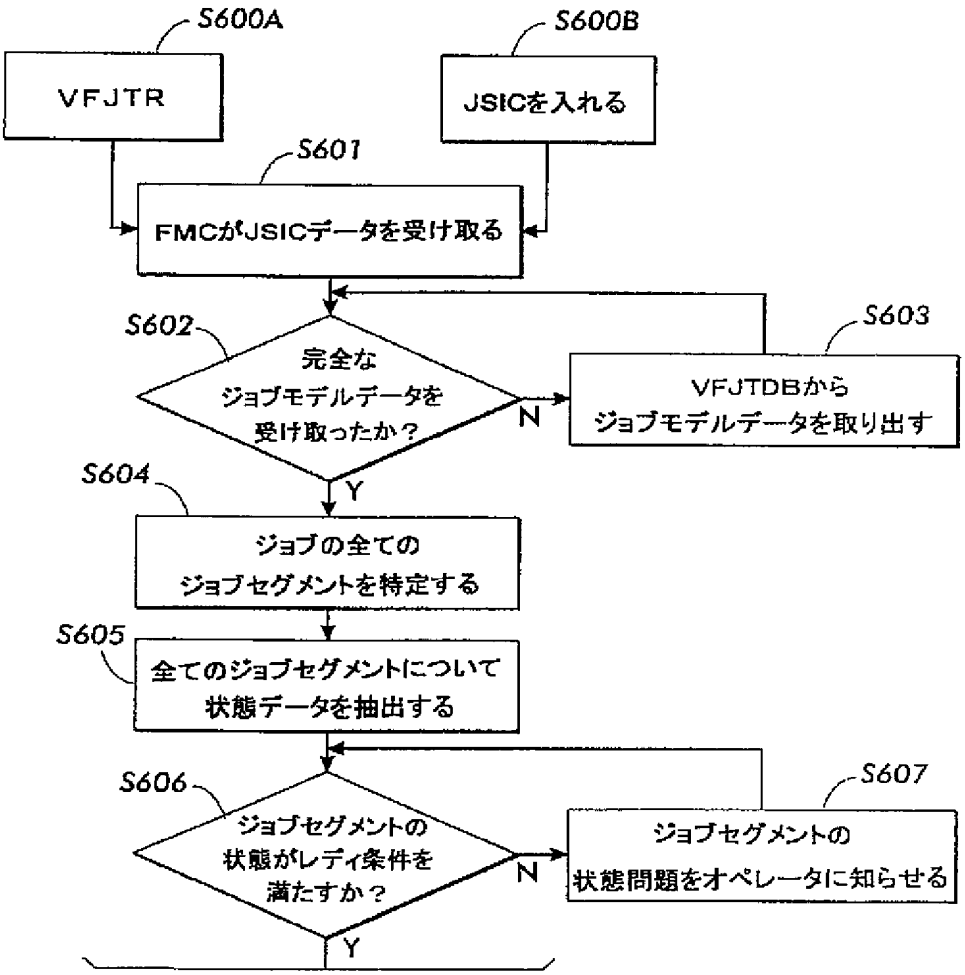

S600A
VFJTR
S600B
JSICを入れる
S601
FMCがJSICデータを受け取る
S602
完全な
ジョブモデルデータを
受け取ったか？
N
S603
VFJTDBから
ジョブモデルデータを取り出す
Y
S604
ジョブの全ての
ジョブセグメントを特定する
S605
全てのジョブセグメントについて
状態データを抽出する
S606
ジョブセグメントの
状態がレディ条件を
満たすか？
N
S607
ジョブセグメントの
状態問題をオペレータに知らせる
Y

図15へ

【図１５】

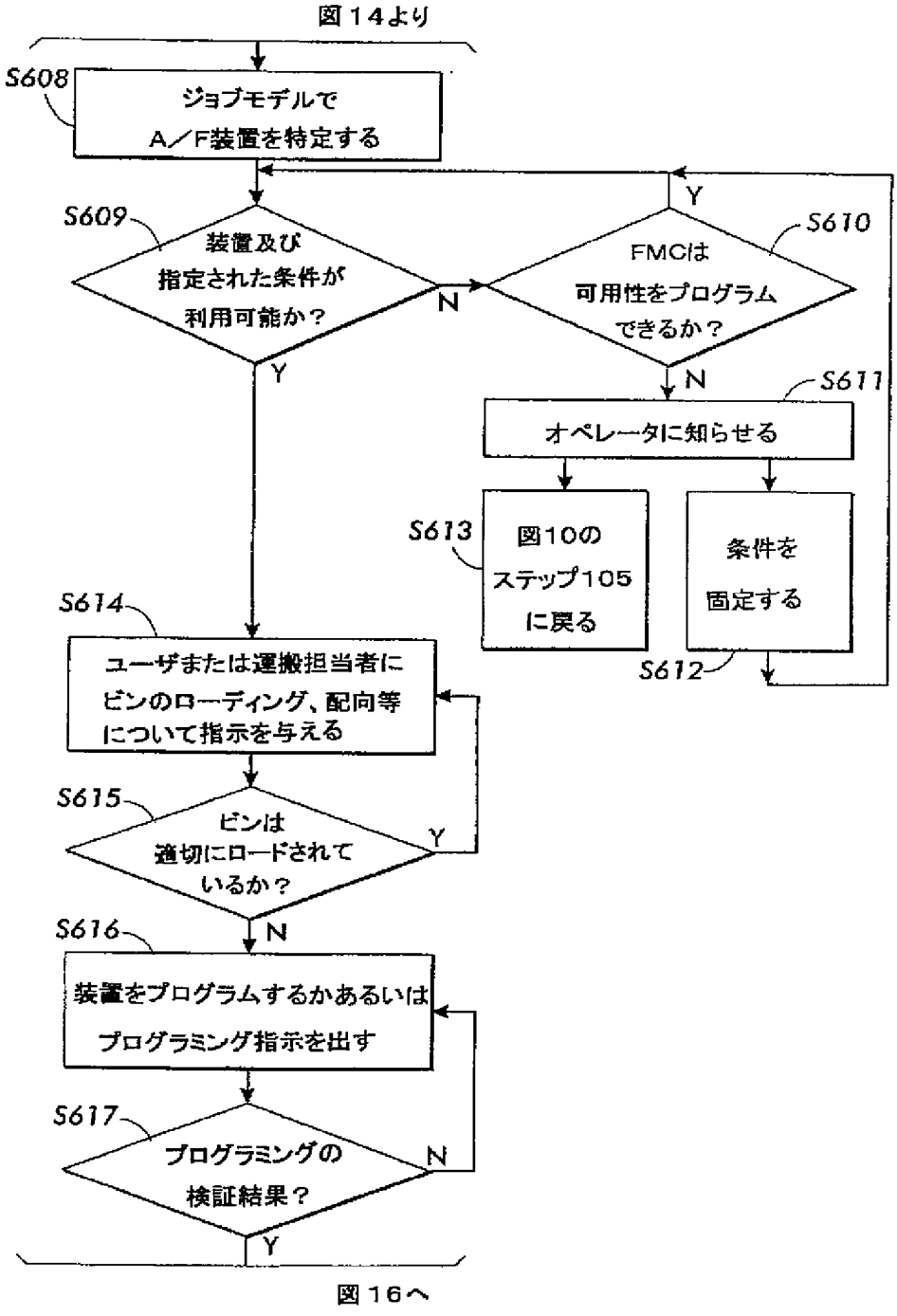


図１４より
S608
ジョブモデルで
A／F装置を特定する
S609
装置及び
指定された条件が
利用可能か？
N
Y
S610
FMCは
可用性をプログラム
できるか？
Y
N
S611
オペレータに知らせる
S613
図10の
ステップ105
に戻る
条件を
固定する
S612
S614
ユーザまたは運搬担当者に
ビンのローディング、配向等
について指示を与える
S615
ビンは
適切にロードされて
いるか？
Y
N
S616
装置をプログラムするかあるいは
プログラミング指示を出す
S617
プログラミングの
検証結果？
N
Y
図１６へ

【図１６】

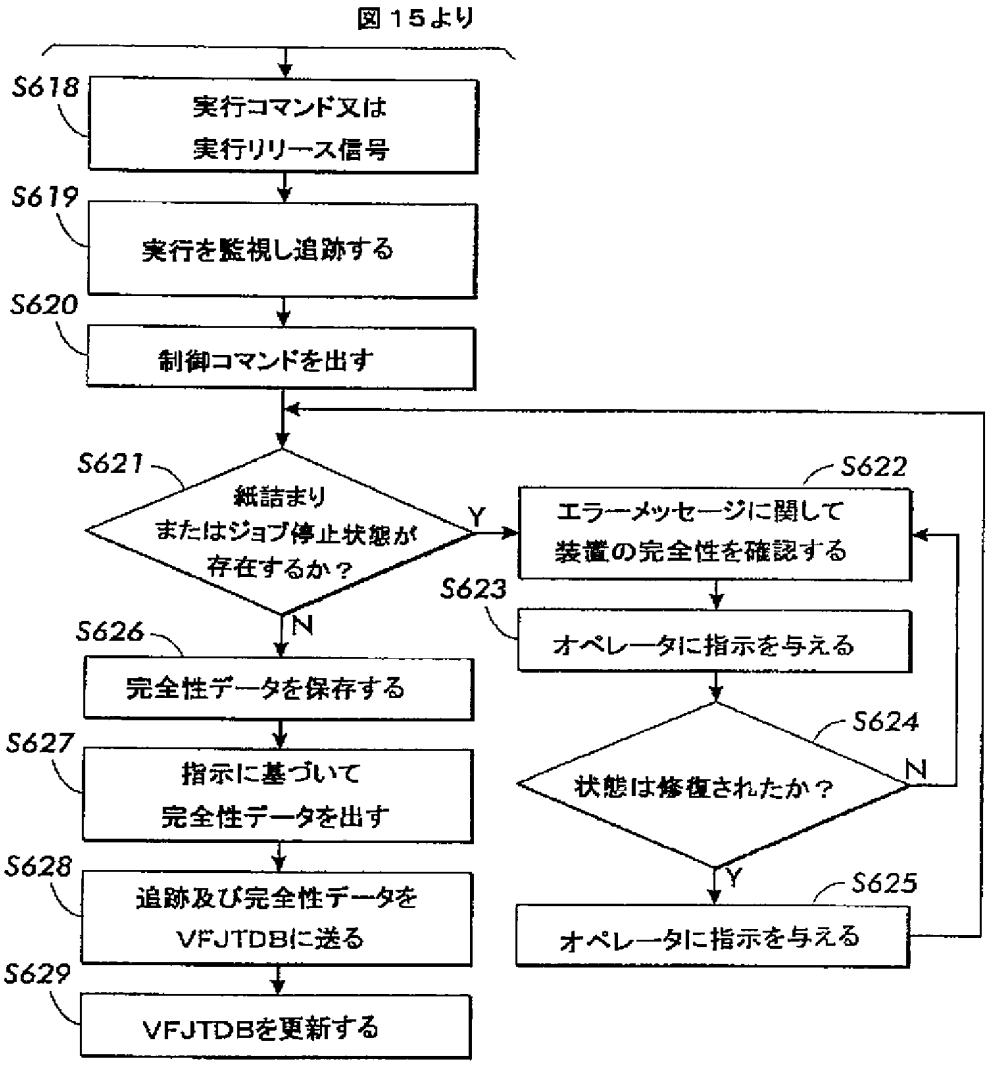
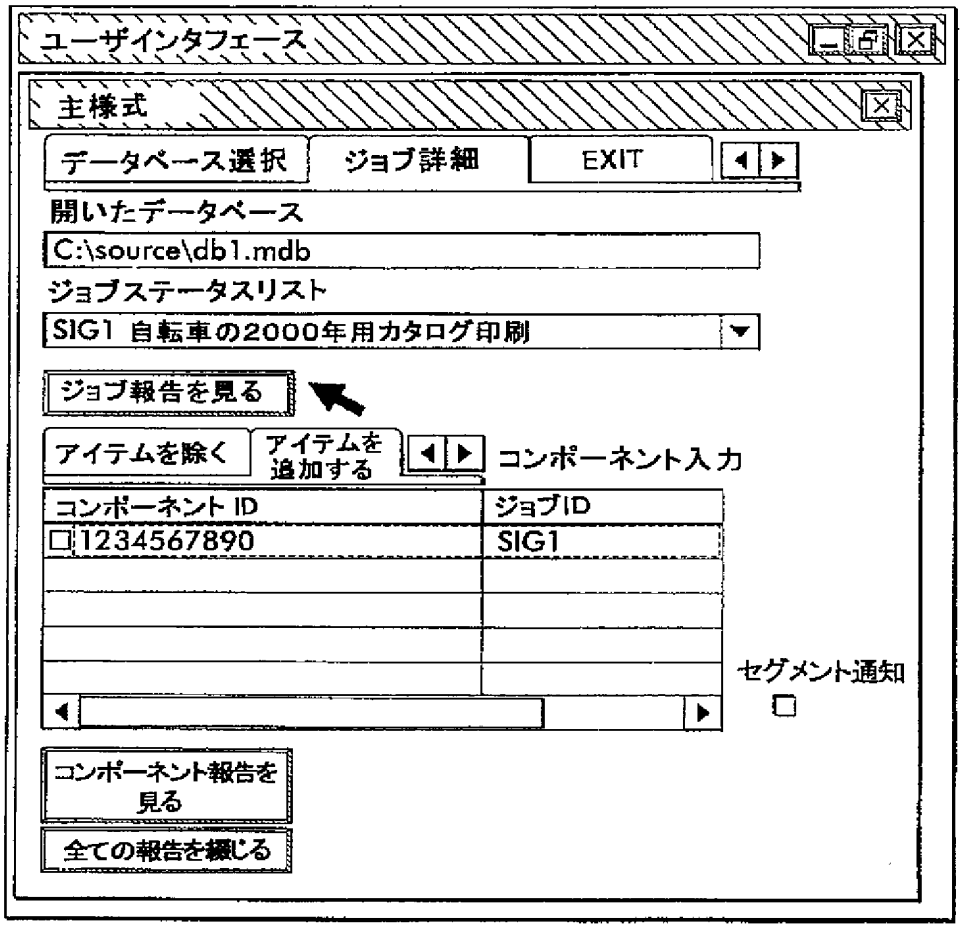
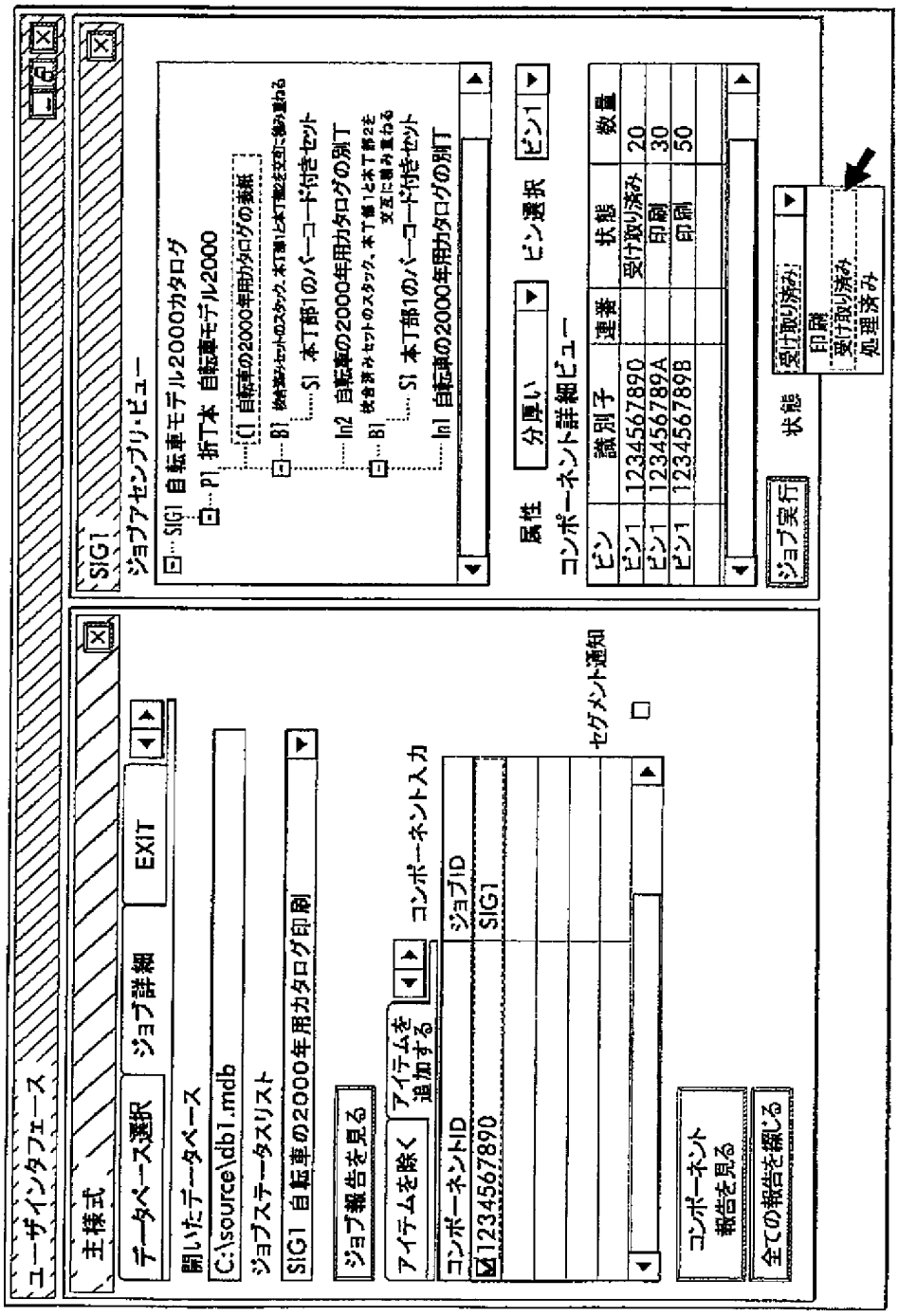


図１５より
S618
実行コマンド又は
実行リリース信号
S619
実行を監視し追跡する
S620
制御コマンドを出す
S621
紙詰まり
またはジョブ停止状態が
存在するか？
Y
N
S622
エラーメッセージに関して
装置の完全性を確認する
S623
オペレータに指示を与える
S624
状態は修復されたか？
N
Y
S625
オペレータに指示を与える
S626
完全性データを保存する
S627
指示に基づいて
完全性データを出す
S628
追跡及び完全性データを
VFJTDBに送る
S629
VFJTDBを更新する

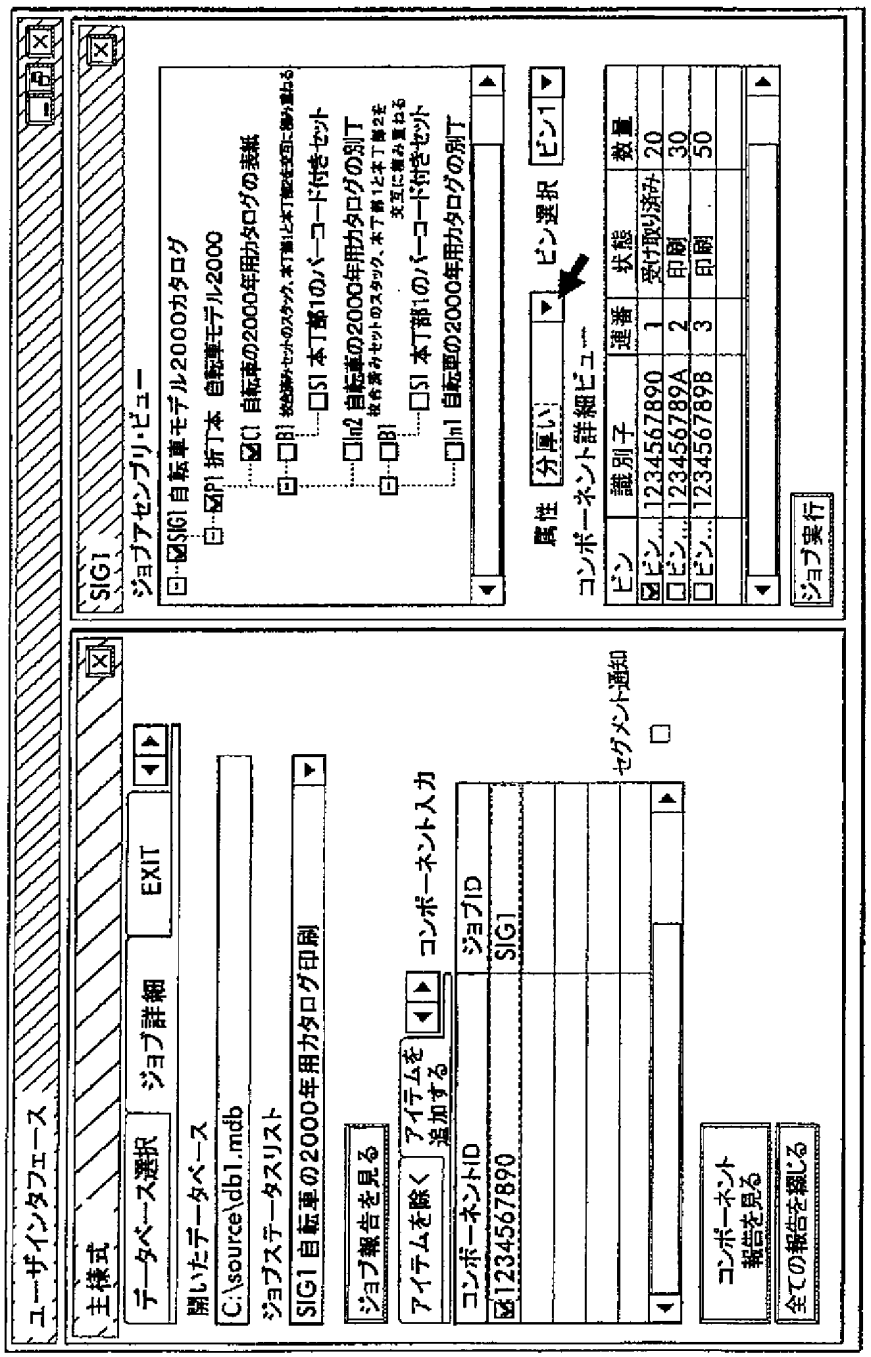
【図１７】


ユーザインタフェース
主様式
データベース選択
ジョブ詳細
EXIT
開いたデータベース
C:\source\db1.mdb
ジョブステータスリスト
SIG1 自転車の2000年用カタログ印刷
ジョブ報告を見る
アイテムを除く
アイテムを追加する
コンポーネント入力
コンポーネント ID
ジョブID
1234567890
SIG1
セグメント通知
コンポーネント報告を見る
全ての報告を閉じる

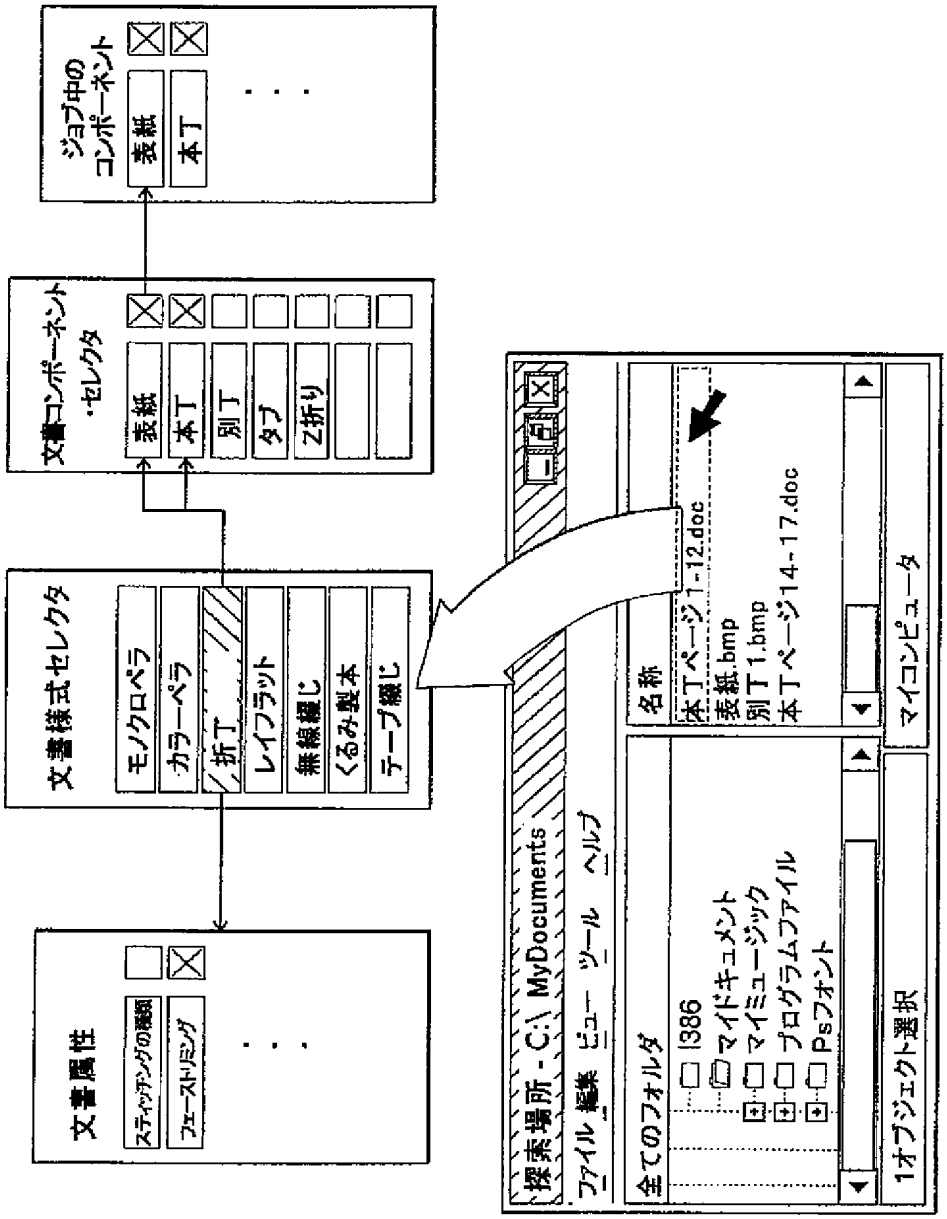
【図１８】


ユーザインタフェース
主様式
データベース選択
ジョブ詳細
EXIT
開いたデータベース
C:\source\db1.mdb
ジョブデータリスト
SIG1 自転車の2000年用カタログ印刷
ジョブ報告を見る
アイテムを除く
アイテムを追加する
コンポーネント入力
コンポーネントID
ジョブID
1234567890
SIG1
セグメント通知
コンポーネント報告を見る
全ての報告を概じる
SIG1
ジョブアセンブリ・ビュー
SIG1 自転車モデル2000カタログ
P1 折丁本 自転車モデル2000
C1 自転車の2000年用カタログの表紙
B1
S1 本丁部1のバーコード付きセット
In2 自転車の2000年用カタログの別丁
B1
S1 本丁部1のバーコード付きセット
In1 自転車の2000年用カタログの別丁
属性
分厚い
ピン選択
ピン1
コンポーネント詳細ビュー
ピン
識別子
連番
状態
数量
ピン1
1234567890
受け取り済み
20
ピン1
1234567891A
印刷
30
ピン1
1234567891B
印刷
50
ジョブ実行
状態
受け取り済み
印刷
受け取り済み
処理済み

【図１９】


ユーザインタフェース
主様式
データベース選択
ジョブ詳細
EXIT
開いたデータベース
C:\source\db1.mdb
ジョブスデータスリスト
SIG1 自転車の2000年用カタログ印刷
ジョブ報告を見る
アイテムを除く
アイテムを追加する
コンポーネント入力
コンポーネントID
1234567890
ジョブID
SIG1
セグメント通知
コンポーネント報告を見る
全ての報告を閉じる
SIG1
ジョブアセンブリ・ビュー
SIG1 自転車モデル2000カタログ
P1 折丁本 自転車モデル2000
C1 自転車の2000年用カタログの表紙
B1
S1 本丁部1のバーコード付きセット
m2 自転車の2000年用カタログの別丁
B1
S1 本丁部1のバーコード付きセット
m1 自転車の2000年用カタログの別丁
属性
分厚い
ピン選択
ピン1
コンポーネント詳細ビュー
ピン
識別子
連番
状態
数量
ピン...
1234567890
1
受け取り済み
20
ピン...
1234567890A
2
印刷
30
ピン...
1234567890B
3
印刷
50
ジョブ実行

【図２０】



フロントページの続き

(31)優先権主張番号　２０４７１６
(32)優先日　平成12年５月16日(2000．5．16)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　２０４７２０
(32)優先日　平成12年５月16日(2000．5．16)
(33)優先権主張国　米国（ＵＳ）

(72)発明者　ヘンリー　ティー．クレマーズ
アメリカ合衆国　14450　ニューヨーク州　フェアポート　カントリー　コーナー　レーン　54

(72)発明者　ケビン　アール．マザーズ
アメリカ合衆国　14428　ニューヨーク州
チャーチビル　ポール　ロード　1355
(72)発明者　ウェイン　アール．スミス
アメリカ合衆国　14534　ニューヨーク州
ピッツフォード　パーク　エーカー　ロード　74
(72)発明者　ジェラード　アール．スターニック
アメリカ合衆国　14622　ニューヨーク州
ロチェスター　セネカ　ロード　590
Ｆターム(参考)　2C020 BA00

【外国語明細書】

APPARATUS AND METHOD FOR DESCRIBING, PLANNING AND AUTOMATICALLY PROGRAMMING COMPLEX FINISHING TASKS

Priority is claimed from United States Provisional Application No. 60/204,624, filed May 16, 2000.

It is believed that the present invention is applicable to the electronic management and control of a wide range of finishing processes characterized by input from multiple production operations and equipment that, depending upon the job, may be variably applied to work pieces that themselves are highly variable between different jobs. Although the present invention is explained in relation to printing and finishing operations for printed documents, the present invention may apply to such industries, without limitation, as include textile production (which may include printing, cutting, sewing, and finishing), packaging operations for various consumer and industrial products, printed wiring board production, etc. In particular, the present invention is applicable to many operations where processes for production of work pieces are managed separately from processes for finishing and packaging of such work pieces.

Creation and production of printed documents often involves many production and finishing operations that are highly variable with each job. In general, the various operations can be grouped into three major phases: 1) creation of the document information, including prepress operations that render the document in a form suitable for printing, 2) printing of the information onto some form of media such as paper, and 3) finishing of the selected media into a completed document. These 3 major phases often have many sub-phases, and the entire process may vary from relatively

simple to extremely complex. The present invention deals with techniques by which a user may provide detailed instructions for each of the three phases such that instructions may be created as early as during the first phase that are sufficient to guide the entire process through to completion of the third phase. Although of potential use in many printing operations, the present invention is particularly applicable to automated systems for creating, printing, and finishing complex documents within a multi-printer, completely digital environment using digital printers.

Traditionally in phase 1, when a document is composed, the person doing the composition will create one or more electronic image files that represent the parts of the document to be produced. These electronic image data files may be stored in many different formats by many different document creation and manipulation programs. For instance, for a complex document such as a book that utilizes color printing for book covers and pictorial inserts, any of a variety of Page Description Languages (PDLs), such as Postscript® and Postscript-compatible languages, may be used to render the color images in printable form. Often different components within a document will utilize different PDLs. For instance, the cover may be created by a different work team or upon different equipment than photographic reprints or other internal color components. Each prepress team or prepress device may utilize a PDL optimized for its use. For pages comprised of simple monochrome text, a desk-top publishing programs may be utilized to render such pages or a simpler word processing language may be utilized. Still other prepress formats may be utilized for printing of inserts, dividers, and other possible components internal to the finished document. There also may be included in the assembly/finishing job non-printed components such as, without limitation, plastic separators, previously printed sheets retrieved from inventory, photographically produced sheets, or specialized media such as vinyl disk holders or perfume sample packs.

Examples of documents with different components and levels of complexity will now be shown by reference to Figures 1-2. Beginning in Figure 1, a simple signature document is shown that comprises an insert component 12 placed face down on a gathering tray or table, followed by body component 10 placed on top of insert 12 which is then followed by cover 11. A finishing operation indicated in block form at F201 is shown. Such finishing operation F201 may comprise simple folding of the signature body or may include center stapling or similar binding operation. When cover 11 is placed on top finishing operation 201 folds the signature, a cover-bound document 21 is created as shown. The completed document 21 is shown to the right of finishing operation F201. For explanatory purposes, the arrangement of components is shown in box form below finished document 21.

Figure 2 shows the result of layering two body components 10a and 10b in a stack with two insert components 12a and 12b in the order indicated. Cover 11 is added last to the stack. Completed document 23 contains the 9 layers expected from such an arrangement, with the middle layer being a double layer comprising insert component 12a.

Obviously, documents may vary greatly in complexity depending upon the number and order of components, finishing options chosen, etc. Typically, various prepress devices create individual components of the document and digitally render these components in formats that are suitable for printing. PDLs such as Postscript™-compatible languages are often used for such purposes. Subsections of the job that require different prepress or printing operations are typically divided by an operator at an early point in the process. After completion of prepress operations for each portion of the job, the operator(s) send the various portions of the job to printers appropriate for each such portion, thereby initiating different "paths" that each portion of the job my take.

Figure 3 shows typical assembler/finisher operations for a moderately complex document. In the shown example, a set of color portions, 30a, 30b, and 30c, have been printed by a color printer and outputted from the printer in non-collated offset form. A set of monochrome portions, 40a and 40b, have also been printed and have been outputted from the printer in a stack of alternating, collated offset sets. After printing and output into their respective intermediate output bins, the various printed sheets have been gathered from their respective printer output bins, transported, and placed in the bins shown in Figure 3 for feeding into the assembler/finisher apparatus. Color components 30a, 30b, and 30c are placed into sheet feeder receiving bins 42a, 42b,and 42c of sheet feeder 42. An example of such sheet feeder equipment integrated coupled with book making equipment is a Model MC80 sheet feeder integrated with book maker Model SPF-20, both manufactured by Horizon International, Inc. Monochrome components 40a and 40b are placed in feeder bin 43a of set feeder 43 in a manner that maintains the alternating, collated offset stack. An example of such set feeder equipment 43 is a DocuFeed 150 sold by Standard Duplicating Machines Corporation, Inc.

It is important to note that in many jobs, receiving feeder bins such as 42a, 42b, 42c, and 43a have stack height constraints that are less than the total stack height of a particular portion of the job that was printed. In the prior art, an operator typically manually separates a stack of printed sheets into smaller stacks that will fit within the constraints of the receiving bins.

Returning to Figure 3, collator 44 is programmed by an operator for interleaving and collating the components in the correct order. When operated, collator 44 operates in conjunction with sheet feeder 42 and set feeder 43 such that various sheets are placed in a completed stack 50 in the correct order within gathering station or gathering bin 45. Next, stack 50 is

delivered to finisher apparatus 46 where it is first folded, The folded signature stack is then bound, trimmed and otherwise finished into a completed document 60. Among the finishing operations that may be performed within finisher 46 are the following: gluing in, adhesive binding, general stitching, saddle stitching, thread sewing, side sewing, stapling, scoring, and trimming.

Much prior art deals with operations that automate tasks internal to each of equipment and processes described above. In particular, much work has been done to provide automatic linkages between prepress operations and digital printing processes, including output from printers at intermediate finishing stations with capabilities such as collating. One aspect of such prior art includes creation of virtual job tickets to electronically convey information from prepress apparatus through to intermediate finishing operations of the selected digital printers. See, e.g., US-A-5,995,721 Issued to Rourke et al. In Rourke et al., for instance, prepress processes examine the attributes of a print job in order to determine which of a variety of printing apparatus are capable of printing each particular portion of the job in accordance with the specified attributes. The instructions governing printing of each specific portion are provided to each printer pursuant to a virtual job ticket. In Rourke and in other prior art, however, digital tracking and control linkages between the paths of various job portions sent to different printers is generally lost after each portion is sent to a different printer. The virtual job ticket is used only during the printing process itself and during any post-printing processes directly linked to the printing phase of the job. Thereafter, the parsed portions of the job are re-integrated not by use of a virtual job ticket providing instructions to offline finishing but by dropping sheets of one parsed portion into "holes" left in the printing queue of a second portion. See Rourke, column 13, line 11-39. Another characteristic found in Rourke and in other prior art is that a job is parsed into portions based upon printing characteristics

and not upon constraints to be encountered during the entire printing and finishing process.

Although two-way digital tracking and control linkages are common within printers that are physically integrated with their own intermediate finishing stations, there are no two-way digital tracking and control linkages between a stand alone printer system and offline assembler/finisher apparatus such as shown in Figure 3 as 42-46. With respect to assembler/finisher systems that are not physically integrated with their respective printers, the following incomplete list of data characterizing a job and the work pieces of the job are useful for programming some or all assembly/finishing operations: the number of sheets of each type; media type; media thickness; orientation of sheets; organization of sheets within each stack of sheets; order of assembling sheets or stacks; operations to be performed; locations for scoring, folding, trimming, cutting, etc; and the type and placement of binding. Many other instructions are often utilized in addition to this basic list of instructions and parameters.

The need in both the prior art and in the present invention is to efficiently convey to and program the appropriate assembler/finisher systems with the above assembly/finishing data, then to track progress of the job through the finishing operations, and finally to maintain integrity of the job in order to detect and/or prevent defective finished documents.

The prior art teaches several methods for accomplishing the above tasks with varying degrees of satisfaction. A very common approach is for a human operator to separately program the assembler/finishing system. Often, the information to program the finishing operation is provided to the operator in a written or printed sheet or set of sheets, called a traveler sheet, that incorporates information describing assembler/finishing operations for all portions of the job. When preparing to load the stack into the finishing equipment, the operator reads the traveler sheet for the relevant

assembly/finishing instructions, including the order in which the components are to be assembled in the finished product. A complete set of attributes for each component is not provided since the skill and experience of the operator enables the operator to select proper bins, parse stacks, orient sheets, and similar tasks when preparing and programming the assembler/finisher device. A more modern version of this same basic system uses a traveler sheet that is encoded with barcodes or other machine readable coded information. When ready, the operator places the traveler sheet before a digital reader which then displays the information to the operator for manual programming. Yet another version of this idea is to place a machine-readable code such as a bar code or glyph on a sheet associated with each job. This sheet may be a cover sheet placed on each stack of sheets. The code is read by a digital reader and its information is used to set up equipment for the job. Even with the ability to encode some job information digitally, the information required in conjunction with moderately complex jobs for programming of off-line assembler/finisher equipment is so complex that in the prior art some manual programming is required.

The complexity of interrelating and automatically programming all of the above printing, assembling, and finishing operations can be understood by contemplating the innumerable parameters and operations that must be coordinated during the course of a reasonably complex printing job, especially one involving multiple printers and multiple finishing operations and assembler/finisher devices. In addition to all of the variables relating to paper selection (e.g. size and type), PDL or other page descriptions, and other parameters and operations applicable from prepress Phase 1 through completion of printing Phase 2, the assembly/finishing Phase 3 requires programming of information that both (1) specifies the complicated details of assembly and finishing operations to be performed upon each sheet or set of sheets within the job and (2) relates each sheet or set of the job to every other

sheet and set of the job. Especially when the sheets are arriving from different sources such as from multiple printers or from both printers and inventory, existing systems have only been able to automate a portion of these programming tasks or have succeeded in performing both of these programming tasks only in very carefully prescribed circumstances.

For instance, in US-A-5,859,711, issued to Barry et al., the problem is discussed in the special case where all of the pages of a document are sent to a job controller as one job and are then divided at page breaks such that each page is treated as a separate print job. By dividing the one job into a plurality of jobs on a page-by-page basis, the various pages can be allocated to a multitude of printers such that one or more printers can be operated as one virtual printer. The advantages include greater speed and the ability to send color pages to color printers and monochrome pages to monochrome printers, thereby optimizing the use of each. In the above manner, Barry teaches a method of optimizing production of a job through the printing Phase 2. Barry does not, however, teach a method for optimizing the assembler/finisher portion of the production or for arranging the printing, separating, and stacking of portions of a job with a view to the capabilities and constraints of the assembler/finisher equipment. At column 18, lines 37-47, Barry provides part of its solution to the problems created when a job has been parsed to different printers and needs to be reassembled: "It is only important that, when the stacks are defined within a given printer, there is some indication, such as a separator page, that will allow the particular stack created between separators, to be assembled with another stack from another printer in the desired print job output." (lines 42-47) At columns 37 and 38, a second portion of Barry's solution is disclosed by teaching that a distributor control can operate to configure and reconfigure the automatic finishing device in response to how a job is divided into stacks, with each stack being treated "as individual entities and queuing them up and processing them

independent of how fast another job stack in a given job is processed through an adjacent print engine." Column 37, lines 12-15. Barry also teaches that a distribution control can configure automatic finishing devices or such devices may be configured by reading instructions printed on each separator sheet in bar code form.

Thus, Barry is concerned with a method of dividing a job into portions that can be routed to different printers for more efficient printing operations. Its finishing teachings are limited to methods for recombining the separated portions of a job back into the correct order for final finishing. There is no teaching concerning how to combine the job portions except how to collate the portions arriving from different stacks. There is no teaching of how to combine printed sheets with non-printed or previously printed sheets taken from inventory. There is no teaching concerning how to break a stack apart except in response to the job portions created for optimized printing. If further divisions of stacks are necessary after printing in order to enable assembly/finishing operations, Barry lacks any method for analyzing or implementing such divisions. Most importantly, Barry does not teach that the capabilities and constraints of assembler/finishing equipment can be used to divide a job into portions.

Perhaps the most complete attempt to provide a structure for automated programming of certain specified types of print jobs is the International Cooperation for Integration of Prepress, Press, and Postpress (CIP3) Print Production Format issued by the Fraunhofer Institute for Computer Graphics. The web address for CIP3 is: http://www.cip3.org. First issued in 1995, CIP3 provides an ability to create a digital "traveler sheet" written in the Postcript language. CIP3 enables a complete digital description of a document and all of its production steps. It is a proposal for a description language, not a control algorithm. As a description language, CIP3 provides one method of relating each page of a job to every other page of the same

job. As described in the Specification of CIP3, the CIP3 format is intended to enable automatic programming of assembler/finisher equipment. What is missing in CIP3, however, is the automatic programming itself. What is also missing is an ability to match described production operations with the capabilities and constraints of the particular equipment available in conjunction with this job. CIP3 is therefore not useful in selecting or optimizing the actual equipment to be used when planning the job. CIP3 also lacks the ability to inter-relate the production effects of one job to a second job, and CIP3 is therefore of limited value in production planning of multiple jobs. Also, by failure to divide a job into subparts that conform with the constraints of the actual equipment to be used, CIP3 is limited in the depth of its ability to relate each sheet of a job to all other sheets that have been prepared as part of the job. Figure 4 shows in schematic form the type of information that can be described using CIP3. Along the Y-axis is a list of the different sheets in this particular job. Note that there is no information concerning how many sheets of each type are printed since CIP3 is concerned with describing the sheets of a single end product and not of the flows necessary to print a production job. The X-axis is a list of the operations that will be performed on each sheet, including in this case, at the intermediate collator step, the merger of the various sheets into stacks. These stacks will then be separately folded and cut and then gathered together, trimmed, and bound. CIP3 is robust enough to associate fairly complex production processes to particular sheets or stacks of sheets (called partial products within CIP3), including such processes as the gathering of multiple partial products of the job and the binding, finishing of such gathered partial products into a stitched or glued document. See Table 3-13, CIP3 Specifications, Version 3.0, June 2, 1998. Even where CIP3 provides an adequate description language for parts of the job, however, it does not provide methods for actual management, control, and tracking of the job while

in production. In other words, CIP3 is intended to help the programming of a job by describing its parts but does not actually perform or direct the programming nor the implementation of the job.

The ability to track and associate the complex data associated with each or stack of sheets within a complicated print job is greatly hindered by the innumerable equipment and processing parameters that characterize and constrain use of each piece of equipment. For instance, every item of equipment shown in Figure 3 may have different paper path constraints or bin-height constraints. Some types of sheet stock may be too thin or too rigid for certain of the printers or assembler/finishers in the system. Some portions of the job may require an intermediate finishing step such as lamination. Specialized equipment such as laminators may require that sheets be delivered and output in a particular orientation, e.g. long edge first, which constraints then require a re-orienting process prior to insertion into the paper path of the next selected piece of equipment. Some portions of the job may require different types of folding, trimming or cutting operation than other portions of the same job. Within a typical commercial print shop, the number and type of constraints affecting selection of different pieces of equipment or combinations of equipment are inherent and innumerable.

The problem of coordinating equipment constraints is further complicated because of inherent mismatches between the output of printers and the constraints of assembler/finisher equipment. For instance, when a stack of sheets from a printer has more sheets than can be received in the assembler/finisher feeder bins, then a human operator or a machine conveyance system typically divides the larger stack by grabbing whatever number of sheets appear to fit within the feeder bin capacity. This means that the various assembler/feeder bins are filled with unequal numbers of sheets. It also means that a stack of printed sheets gets divided and separated. The separated stacks are typically stored in intermediate bins and must be stored,

tracked, and retrieved when needed. Even where large stacks of sheets are divided by a manual or automated counting procedure such that each separated stack is maintained with a known quantity of sheets, there is no ability in the prior art to dynamically adjust the size of such stacks to conform to the varying constraints affecting each particular printing job and the particular equipment selected for that job. Moreover, stacks of pre-counted sheets are of little use when responding to sheets that are found defective, missing, or damaged. Lastly, where a job can be routed to one of a variety of assembler/finisher systems, each having different constraints, it would be desirable to dynamically change the stacks of printed sheets in order to optimally conform to the constraints of the particular assembler/finisher equipment that becomes selected. This would be especially desirable when managing queues of finishing jobs in systems that are capable of rerouting assembler/finisher operations in response to equipment breakage or unavailability due to use of initially selected equipment with other jobs.

As described above, another task required during the print production process is effective tracking and integrity checking. Within individual printer systems, these tasks are well understood. For instance, in typical production printing systems such as the Docutech® Model 6155 marketed by Xerox Corporation, sheets are counted when fed from the system input feeder and are timed or recounted during or after every major operation performed within the machine. In the event that a sheet does not arrive at a designated station within the time constraints permitted by the system or is otherwise detected as missing, a jam is declared and the portion of the machine apparatus operating prior to the jam is paused. An operator is then directed to clear the jam by removing all sheets residing in the paper path. Since the system controller has tracked and counted each sheet, it knows the number and image file identity for each sheet that has been removed. When the system detects that all sheets have been cleared, the

controller directs the re-imaging and processing of each of the removed sheets. For duplicator systems with recirculating document feeders (RDFs), the same concepts apply except that the controller directs the RDF to circulate the input document pages until the first missing document page returns to the top of the imaging queue.

For print jobs requiring use of multiple printers, the above tracking and integrity functions are less completely managed in the prior art. For instance, in the state-of-the-art Book Factory® system launched by Xerox Corporation in 1999, a primary production printer based upon a Docutech Model 6180 printer system is physically integrated with assembler/finisher apparatus capable of any of the following finishing operations plus appropriate combinations of these operations: Collecting, Folding, Trimming and Adhesive Binding. The Book Factory system also comprises an assembler with a manually loaded input tray for accepting inventory sheets, including covers, printed or prepared by other printers or processes. Although the Book Factory assembler/finisher apparatus can count and time the progress of sheets in much the same way as described above in relation to single printer systems, there is no two-way communication concerning job status to the printer or inventory systems that supplied the manually input sheets. Thus, in the event a jam is declared, there is no method by which other printer systems may automatically be directed to reprint the removed sheets. In normal operations, operators prepare for missing or destroyed sheets by printing extra copies. Where extra copies are not available, then an operator must reprogram the other printer system and reprint the job. In any event, the result is typically waste of extra or removed sheets, consumption of extra consumables such as paper and ink, and expenditure of valuable time by operators and expensive printers.

In sum what is missing in the prior art is:

1) an integrated digital architecture for interactive control, tracking and integrity functions of all three phases of the prepress/printing/finishing process;

2) an integrated apparatus and method for enabling an operator, prior to printing of a job, to provide complete instructions for complex assembler/finishing operations, especially if off-line from the printer controller;

3) a method for accurately and completely describing the final document form of a complex document in a manner that enables both printer controllers and assembler/finisher controllers to instruct and control their respective printing and assembling/finishing operations;

4) an apparatus and method for dividing and managing a print job and associated work flow in response to constraints of both the available printing systems and the available finishing systems;

5) a queue management system that manages the entire print and finishing process in accordance with the availability of specified printers and of specified assembler/finisher equipment;

6) an interactive integrity check system capable of tracking each sheet through each production process, especially those portions of the process involving assembler/finisher operations that are offline from the printer controller; and

7) apparatus and method, acting in response to an interactive integrity check function, that automatically instructs printers to add or replace sheets not available to the assembler/finisher apparatus when such assembler/finisher apparatus calls for such sheets.

For purposes of the present invention, the following definitions shall apply:

A "document component " shall mean a collection of one or more sequential sheets of media that have similar qualities or characteristics

and thus would be printed or non-printed and would be finished or produced in a similar manner. Examples of document component types are covers, bodies and inserts. When collected together in a specific order, a collection of document components may form a complete "document". Each document component may require its own intermediate finishing operation before its final assembly/finishing into a "document". For instance, a cover sheet may require lamination in an intermediate finishing process before conveyance to the final assembly/finishing apparatus.

A "document" shall mean a collection of one or more document components placed in a specific order and finished in a manner that relates the various document components to each other.

A "document form" shall refer to the manner in which the various components are finished into a composite form, including such operations as folding, cutting, stitching, binding, and gluing. Each document form requires unique image imposition, printing, finishing process requirements and physical identifying characteristics. The structure of most typical documents can be classified into one of 7 standard "document forms":

Flats: One or more sheets of media, unfolded, that are either not bound together or are bound by means of a device such as a staple or a wire that penetrates the media from one side for the first sheet to the other side of the last sheet.

Signature: One or more sheets of media that are folded. The sheets may or may not be stitched together but if they are, they are stitched on fold from outside to inside. The images are imposed on the printed sheets of media such that after folding the images on the sheets from pages that are in the correct order for reading (readers spread).

Perfect Bound: Individual sheets of media grouped together or gathered signatures that are commonly glue-bound to a flexible, wrap-around cover which protects the body/contents. The edges of the book-block and cover are usually flush with each other.

Case Bound: Individual sheets of media grouped together or gathered signatures that are commonly glue-bound to a rigid, wrap-around cover

which protects the body/contents. The edges of the cover usually extend past the book-block.

Lay Flats: Individual sheets of media with holes or notches along one edge grouped together and commonly bound to a separate front and back cover which protects the body/contents. Common binding methods include wire-coil, plastic-coil, plastic-comb, 3, 5, and 9 ring binders, etc. The document lays flat and does not close when opened up. The edges of the cover usually are flush with the book-block.

Tape Bound: Individual sheets of media grouped together and commonly bound to a front and back cover which protects the body/contents. The binding method is an adhesive tape that wraps around the book-block overlapping the front cover and back covers along the spine edge. The document usually will not lay flat when opened up. The edges of the cover usually are flush with the book-block.

Other document forms can be specified as needed.

A "constraint" shall mean a limitation of a device based upon its design or use. A "constraint" may be permanent or temporary. Examples of permanent constraints would be inflexible bin heights or widths, temperature limits for laminators, bin type (set feeder or sheet feeder), method of feed (e.g., top or bottom feeder), required order (n-to-1 or 1-to-n), face up, face down, required orientation (e.g., leading edge must be long or short dimension), paper path width, thickness for folding and trimming, transformations enabled within the device (e.g., face up to face down, lead-edge reversal to trial-edge), landscape to portrait orientation, etc.) and similar limits related to a device's design. Examples of temporary constraints include: a period of time that a piece of equipment or part of equipment, such as a particular bin, is not available due to a broken part or use for another job; or (b) the type of media, glue, binder material, etc., then available for use within a particular piece of equipment.

A "job segment" shall mean a stack of sheets produced by a common printing or finishing process and conforming to the same printing and

finishing constraints. A "job segment" may contain a single document component, a portion of a large document component, or a collation of several document components. As will be explained below, "job segments" are identified in order that document components with similar printing and/or finishing requirements are grouped together for efficient printing, handling, and finishing. For example, if a document has two 8.5" X 11" monochrome body components, both body components may be grouped in the same job segment in order that they will be printed on the same printer at the same time. Depending upon requirements, these components may be output at the printer as collated or non-collated stacks and, if the components are collated, the collated stacks may be placed in an offset manner in order to indicate separation between the collated sets. As another example that is particularly pertinent to the present invention, if an input bin of the selected finishing apparatus has a stack height constraint of 2.2 inches, then the maximum stack height of a "job segment" will be 2.2 inches even if the total stack height of a particular document component or of a collated stack of components is much higher. For this situation, a "job segment" during printing Phase 2 may comprise all of the sheets that are printed at the same printer. Within this large job segment stack, however, smaller "job segments" limited to 2.2 inches in height may be separated in an offset manner or separated by separator sheets. Thus, segmentation of a job would be done based upon an offline finishing constraint that does not otherwise affect the operations of the printer system.

As used herein, "finisher" and "assembler/finisher" shall both refer to systems designed to perform assembly and/or finishing operations.

A "node" means each unit of a job arranged within a hierarchy of units, i.e., the job itself, each document within the job, each document component within each document, each stack, sheet, set of sheets, etc. The hierarchy descriptors found in CIP3 can be useful in identifying many of the

"nodes" of a job. When used in conjunction with object-oriented software, a "node" may be treated as an "object" that can moved or manipulated by itself or may be opened into subparts that themselves are objects that can be moved or manipulated.

One aspect of the present invention is a software architecture by which the assembly and finishing Phase 3 of a complex document can be managed as early as during the initial Phase 1 set up of the job by use of an architecture in which the document is represented as a series of individual document components that, when assembled in a specific order, can be classified in one of the specified document forms.

Turning now to Figure 5, an overview of this aspect of the present invention is shown. Figure 5 contains a block diagram showing the flow of work using the present invention and showing some of the relationships between various items of equipment using the present invention. Within Figure 5, box 1 represents the prepress operations of Phase 1. The output of the prepress operations of box 1 is a set of appropriate PDL files that are delivered to a Production Monitor Controller (PMC), 100. As described more fully below, the PMC is a controller that coordinates overall production of the print job.

Figure 6 shows the typical inputs and outputs of a PMC 100 in block diagram form, including the relationship between the PMC and a Virtual Finishing Job Ticket Database "(VFJTDB)", 501, to be described below. In general, the inputs to the PMC 100 include some or all of: 1) from the Virtual Printer Job Ticket Database ("VPJTDB") (described below)VJTDB, a list of printer capabilities and constraints; 2) from the VFJTDB, a list of assembler/finisher capabilities and constraints; 3) a description of the finished product which may be a CIP3 or similar description; 4) PDLs and other files for the content to of each sheet to be printed; 5) production information such as the number of copies, targeted printing devices, and any special finishing

or packaging attributes, including, without limitation, the identity and retrieval location of any non-printed and/or inventory items. In general, the output from the PMC includes identification of each job segment for each operation within the job as well as a complete set of Phase 2 printing and Phase 3 assembling/finishing instructions for each job segment. More specifically, the output from a PMC comprises some or all of: 1) a job segment descriptions and identifiers for each job segment; 2) a database representation (such as the VJTDB description explained below) of the structure of the job segments and the document components, sheets or sets within the job segment; 3) a PDL file for a job tracking sheet, if any; 4) a PDL for a fetch sheet, if any; 5) integrity descriptors encoded into the VFJTDB for later use by a Finishing Module Coordinator (FMC); 6) virtual job tickets for printers and Assembler/finishers; and 7) a prompt to call for one or more human operator responses. A more detailed description of the processes within a PMC is provided below in relation to Figures 7-10.

Returning to Figure 5, the instruction set for Phase 2 printing and Phase 3 assembling/finishing is output from PMC 100 in the form of both a Virtual Print Job Ticket (VPJT), 101, and a Virtual Finishing Job Ticket (VFJT), 102. The VFJT and VPJT may contain the complete instruction set for the job or may simply contain reference pointers to a database where such information is retained. The VPJT, 101, is conventional in the art as discussed above in relation to US-A-5,995,721 issued to Rourke et al.; US-A-5,615,015 issued to Krist et al.; US-A-5,760,775 issued to Sklot. The details of a VFJT, 102, and the method of its creation by a PMC, 100, will be described in more detail below.

The data for each VFJT is recorded by the PMC in a Virtual Finishing Job Ticket Database (VFJTDB), shown in Figure 5 as 501. A VFJTDB is a database or a data file that contains all job construction, control and integrity data necessary to take the prints coming from the printing

device(s) and perform the necessary finishing processes to turn the prints into the desired final document form. The format of the VFJTDB can be hard copy (print), soft copy (floppy, CD-R, CR-RW) or electronic (electronically stored in memory or on a hard disk drive) copy form. It could be either human or machine-readable or both.

The type of data and instructions required in a VFJTDB 501 for each job are information such as but not limited to: accounting and administration information, sheet, set and job level finishing instructions, color and print quality control data, registration data, etc. The data and instructions also contain a description of the job segments (stacks and stacks of sets) of the job being produced and instructions on how to reassemble these pieces to complete the processing of the job. Additionally this information can enable the automatic setup of the finishing device(s), integrity control and monitoring throughout the full scope of the production processes. The VFTDB provides the basis for a direct link between the offline finishing operations and the integrity control functions of online printing and intermediate finishing systems. The VFJTDB data can take on the form of a proprietary format or an industry standard format such as but not limited to a modified form of CIP3. More information concerning the structure and use of the VFJTDB 501 is contained below.

Returning now to Figure 5, Phase 2 of the printing process is commenced after delivery of the VPJT, 101, to one or more Digital Front End Print Controllers (DFE) represented by box 200. Such DFE's are conventional in the art. Examples include PDL products made by Splash, Harlequin, Adobe, and others. In conformance with instructions provided in the VPJT, 101, the print job is divided into separate printing job segments and is distributed to various print engines for printing using the printer or press which the operator or PMC, 100, believed to be optimal when the VPJT was first established. Alternatively, the VPJT may provide that the DFE, 200,

sometimes through interaction with the PMC, 100, may automatically select the appropriate printing device based upon dynamic queue and print selection criteria.

Boxes 201-204 of Figure 5 are examples of various types of printers to which document components may be delivered for printing. Printer 201 is a cut sheet digital printer connected to an integrated finishing module 201A. Integration between printer 201 and finisher module 201A is accomplished using DAF or MFA-type protocols. As discussed above, a typical finisher module 201A includes capabilities such as collation, folding, and simple binding such as stapling. Printer 202 is a cut sheet printer with a combination of color and monochrome printing capability. The Document Centre® Color Series 50 printer sold by Xerox Corporation is such a printer. Finisher module 202A is integrated with printer 202 as shown in Figure 5 and may have capabilities similar to that described in connection with finisher 201A. Similarly, printer 203 is shown as a continuous form feed printer and is integrated with finisher module 203A. Printer 204 represents the various apparatus and processes normally associated with offset printing, including the prepress steps of preparing offset plates at a plate imager 204A, plate developer processor 204B, and offset printing press 204C. Unlike printers 201-203 which may be digitally integrated with their respective intermediate finishing modules 201A-203A, offset presses are not digital imaging devices and lack direct digital integration with assembly and finishing equipment.

As shown in Figure 5, each of finishing modules 201A-203A and offset press 204C place their respective job segments in their respective output trays or bins 201B-203B and 204D. When placed in such trays or bins, the job segments or may not be collated, stacked or otherwise separated for handling and conveyance. Also as described above, each of finishing modules 201A-203A may provide some intermediate level of finishing such as folding or stapling. Multiple document components may be printed or

assembled at the same printer and intermediate finishing station and be treated during this phase of the job as one job segment. Conversely, a single large document component may be output in a stack with separator sheets or offset stacks indicating multiple job segments within the single document component.

Another aspect of the present invention is the association of a unique Job Segment Identifier (JSI) with each job segment. In Figure 5, a sheet containing a JSI is shown in association with each job segment that is output from printers 201-204. The respective JSI sheets are labeled 201C-203C and 204E, respectively. For complex jobs or for document components that are printed in large stacks, there may be many JSIs corresponding to many job segments within the job or within the stacks.

A JSI can assume any form that can be associated with a job segment throughout the finishing and other applicable printing processes. Among such forms are copies stored in (a) a printed sheet printed and placed on top of a printed job segment, (b) system memory such as hard drives, (c) magnetic media such as floppy disks or magnetic strips, (d) optical memory such as CD-ROM or CR-RW disks, (e) bar code symbols printed on sheets associated with the Job Segment, or (f) any other means by which machine or human readable identifying information may be associated with a Job Segment. A JSI may be machine, human readable, or both depending upon the phase of the job. Indeed, in the event that a scanner is capable of reading the top printed page of a job segment in such manner that the job segment can be uniquely identified, then no special symbols or special top page would be necessary. Thus, each JSI contains, at a minimum, a job and job segment number or other identifier that uniquely identifies the job segment from all other job segments. Typically, the JSI comprises both a unique job number and a Job Segment Identifier Code (JSIC). The job number uniquely identifies the print job from all other print jobs and the JSIC uniquely identifies the job

segment. In one embodiment, the JSIC comprises recognizable unique text on the top sheet of a job segment, which JSIC forms a vector to a JSI that remains encoded in digital memory. Whichever form a JSI takes, the JSI serves as a reference pointer to the portion of the VFJTDB that describes the contents of the identified job segment. The JSI remains associated with the applicable job segment when it is transported from the printing device(s) to other finishing processes. This enables tracking of the job segment from the printing device(s) to the assembler/finisher apparatus. Whether or not the job segments are part of a job that requires prints to be produced on one or more printing device(s), each JSI will have a common job number but a different JSIC that uniquely identifies each particular job segment of the job.

Returning to Figure 5, the JSIs are shown in the form of a printed sheet called a Job Segment Identifier Sheet (JSIS) that is typically printed along with the sheets of the job and is placed on top of the job segment stack in the output trays or bins, 201B-203B and 204D. Such JSIS sheets are shown in Figure 5 as 201C-203C and 204E. Information on a JSIS comprises either (a) a pointer (the job number and JSIC) to a Virtual Finish Job Ticket Database (VFJTDB) stored in some other electronic or soft copy format or (b) the portion of the VFJTDB itself that provides instructions for the job. Such instructions may be printed on the JSIS in electronic or human readable form. In contrast to conventional separator sheets that are placed upon each stack of printed output no matter how large the stack, each JSI serves as a unique identifier of each job segment of a print job. An example of a JSIS is shown in Figure 7. Human readable text comprising the JSI and job instructions is shown at region 601. In region 602, machine readable glyphs are shown containing the full data content of the VFJTDB applicable to the identified job segment. In region 603, a machine readable bar code is shown which comprises a pointer to the VFJTDB stored elsewhere. Once all

sheets of a job and all JSISs have been placed in the output bins or trays, Phase 2 of the printing process is complete.

Phase 3 of the printing process comprises the final assembly and finishing phase wherein the various document components are gathered from output trays or bins 201B-203B and 204D, assembled in a particular order, and finished into a specified document form. In the prior art, where multiple printing devices are used, operators configure and operate these Phase 3 steps separately from operations performed in each of Phases 1 and 2. Only in those instances in which all of the assembly and finishing is accomplished within a single digital printer and governed by a unified print/finisher controller does the prior art teach that Phase 3 can be configured and controlled automatically. In Figure 5. arrows 301 and 302A, B, and C show the conveyance of printed job segments from output trays or bins 201B-203B and 204D to finishing Set Feeder Module 402 and Sheet Feeder Module 401, respectively. In conventional systems, such conveyance is often manual although automated conveyance systems are used in certain applications. Even when automated conveyance systems are utilized, the prior art does not teach a method by which offline assembler/finisher equipment may be programmed to automatically process a complex assembly and finishing operation based upon instructions created prior to printing of the sheets.

In the present invention, each job segment arrives at the assembler/finisher apparatus with a JSI reference pointer. As noted above, this typically will appear on a JSIS although any form of JSI will suffice. The purpose of the JSI is to identify a particular job segment to a Finishing Module Coordinator (FMC), 700, which is a controller of the present invention that directs the assembler/finisher operations. In Figure 5, a Virtual Finishing Job Ticket Reader (VFJTR) is shown as 701 and is responsible for reading the JSIS or for otherwise providing information to the FMC, 700, sufficient for the FMC to determine the unique JSIC. Humans may also intervene in the

process to submit JSIC's to the FMC, particularly if a JSIS is only human readable. The FMC, 700, is a software-based controller that manages, interprets, sequences, and allocates assembler/finisher production data. Using a variety of interfaces to each assembler/finisher device, the FMC communicates to each device the data required to program that device for implementation of the job. It tracks each job segment through the process and ensures that job segments are properly loaded before the devices begin operating. The FMC also typically provides information to human operators concerning job status and in order to enable operators to make production decisions where necessary or appropriate. The FMC operates by receiving the JSI that identifies each job segment and determining whether the JSI itself contains all required assembler/finisher data. If a JSIS or similar JSI does not provide all instructions for finishing the job, then the FMC uses the JSIC to retrieve all relevant information concerning the job model stored in the VFJTDB. The FMC then reviews the assembler/finisher combinations prepared by the PMC to ensure that all identified devices are currently available. Once this condition is satisfied, then the FMC determines the bins or other assembler/finishing locations where each job segment should be placed. In general, the FMC communicates with the PMC through the VFJTDB. Where assembler/finisher devices are automatically programmable, the FMC typically is programmed to interact with the specified interface format for each device in order to automatically provide programming instructions. Job tracking and integrity information would also be provided. When all required job segments have been loaded in their appropriate bins, the FMC would either direct the assembler/finisher devices to begin or would inform human operators that the job is ready. In this manner, the complete assembler/finisher operation can be controlled, implemented, tracked, and checked for integrity. More details concerning the design and operation of the FMC is provided below. For purposes of this invention, it is important to note

that the functions of the PMC and FMC are described as separate controller functions. It is possible in the present invention for these controllers to be combined or for some functions described in relation to one controller to be reallocated to the other controller.

Details of the PMC, shown as Box 100 on Figure 5, will now be described in relation to Figures 8-11. Figure 8 provides an outline of one embodiment of a logic architecture for the PMC of the present invention. In the embodiment shown in Figure 8, the capabilities and constraints of the various devices used in a job are used to help plan the job after PDLs or similar content files have been created. Alternative embodiments of the present invention may use data concerning capabilities and constraints to help create PDLs, imposition files, and similar content and layout files.

Each of the steps shown in Figure 8 will be described in more detail below in relation to Figures 9-11. Within Figure 8, beginning at step 70, a process for creating a job model file is implemented. Construction of a job model using the present invention typically begins after PDL files or other page description, imposition, and similar page content and layout files for the job have been completed. In one embodiment, construction of the job model begins after the job has been laid out using CIP3 or similar job layout format. At step 70, a job model file is opened by assigning at least a job name, a job identifier code, and an identifier code to a file location in a database.

At step 71, attributes are assigned automatically or by the user to each "node" identified in the high level job model received by the PMC. "Node" is defined above to mean each unit of a job arranged within a hierarchy of units, i.e., the job itself, each document within the job, each document component within each document, each stack, sheet, set of sheets, etc. The hierarchy descriptors found in CIP3 can be useful in identifying many of the "nodes" of a job. When used in conjunction with object-oriented software, a "node" may be treated as an "object" that can moved or

manipulated by itself or may be opened into subparts that themselves are objects that can be moved or manipulated. At step 71, the identifiable nodes are typically at the job, document, and document component levels.

At step 72, the operations to be performed on each node are identified. Again, CIP3 may be a useful tool for identification of certain operations. Next at step 73, source files for each of the document components are assigned. Next, at step 74, printers that are available to output the job are assigned. At step 75, finisher/assembler devices available for the job are assigned. At step 76, the PMC evaluates whether intermediate finishing operations such as collation should be assigned to printer systems having such capabilities or whether such operations should be performed by non-integrated assembler/finisher equipment. At step 77, the key step of generating job segments occurs. As will be explained in more detail below, job segments for each operation of the job are determined based upon the attributes and operations associated with each document component plus the capabilities and constraints of the various printers and assembler/finishers that may be used for the job. At step 78, various outputs from the PMC are shown. At step 78A, instructions for printing and/or other preparation of an intermediate job segment is shown. An example of such an intermediate job segment might be the initial printing of JSIS or other sheets that will ultimately run through two or more printers. After such JSIC or other intermediate preparation, the initial job segment may be broken apart or combined into different job segments that are appropriate for the next printing or assembler/finisher step. At step 78B, instructions for final printing of each job segment are provided. At step 78C, a copy of the job model and job segment descriptions is sent to the Finishing Module Coordinator (FMC) via the VFJTDB (boxes 700 and 501, respectively, in Figure 5). For each of steps 78A and B, instructions for creating Job Segment Identifier Sheets (JSIS) or other job segment identifiers associated with each job segment are typically

sent. Within step 7C8, copies of the JSIS or other identifiers are typically made directly or indirectly available to the FMC.

Turning now to Figures 9-11, more details are provided for each of steps 70-78 shown in Figure 8. Figure 9 shows one embodiment of the portion of the PMC that creates a job model and roughly corresponds to steps 70-73 of Figure 8. Figure 10 shows one embodiment of the portion of the PMC that assigns various printer and assembler/finisher devices to implementation of the job and arranges the order of such operations and roughly corresponds to steps 74-76 of Figure 8. Figure 11 shows one embodiment of the portion of the PMC that creates job segments for each job and roughly corresponds to steps 77 and 78a-c of Figure 8.

Turning now to Figure 9, the sequence begins with step 80 which represents all of the prepress steps involved in the document creation, manipulation and imposition of each sheet of the job. Among the possible outputs from step 80 are the following types of files: PDL files, imposition files such those created by PREPS imposition software available from ScenicSoft, Inc., CIP3 files, files created by DigiPath® production management software available from Xerox Corporation, Word, Powerpoint, Photoshop, and any number of other prepress creation and preparation software. In general, each of these files operate on single sheets or single document components. In step 81, a human operator originates the job creation process by selecting the database in which a job will be stored and, consistent with the rules applicable to such database, assigns or creates a job name and a job number. This data constitutes a job node, and at step 81, such job node is added to the database. At step 82, the user selects a document form from a menu of available document forms and adds this data as a document node entered under the job node file in the database . At step 83, the user adds document form attributes that are selected from a menu of attributes that are specific to the selected document form. These attributes are added to the document

node file in the database. At step 84, the user names a document component to be added or modified as a component node filed at the level below the document node in the database. At step 85, the user associates particular PDL or other content files with the component node in the database. Examples of content files include those described above in relation to step 80. At step 86, attributes applicable to each document component are assigned to the applicable document component node. "Attributes" in this context means such descriptors as the type or color of paper, imposition information, and any other descriptive directions pertaining to the processing of that particular document component. At step 87, the PMC checks to confirm that the component descriptors, PDLs, and attributes all conform to "Document Component Form Rules" for the applicable document form. For example, a document component form rule for a signature form may provide that only the center sheet of a signature can be associated with a PDL that covers a 2-up sheet without a gutter in the middle. Document component form rules may also check to ensure that locations of folds occur in gutter regions. If the check in step 87 detects violations of document component form rules, then the operator is notified and the process is returned to step 84. Assuming that no violations are detected, then at step 88, the assignment of attributes to the each document component is deemed complete.

At step 89, the user is asked whether an additional document component will be added to the job. If yes, the user interface returns the user to step 84. If not, the PMC algorithm proceeds to step 90 where a "Document Form Rule" operator compares attributes of all the document components to each other and to a list of attributes that are either specifically permitted or prohibited for components within the selected document form. For instance, if the imposition attributes of the various signatures are not consistent with each other (e.g., the gutters and fold locations are not aligned), then the "Document Form Rule" operator 90 will notify the user at step 91 of a violation of the

document form rules and will prevent finalization of the job model until corrected or overridden by the operator. If the document and its document components all conform to their respective Document Form Rules, then the job is passed to step 92, where the document model is deemed complete. If instead a violation of Document Forms Rules is detected, the user is returned to step 89 where he/she is asked to make appropriate corrections via a return to step 84. When a job has been passed to step 92, the user is interrogated, at step 93, whether another document is to be added to the job. If yes, then the user is returned to step 82. If not, processing continues to step 94, where the job model is deemed complete.

Turning now to Figure 10, the portion of the PMC that selects devices and sequences for implementation of the job is shown. Beginning at step 100, the job model is received from the conclusion of the processes shown in Figure 9 and is filed in the job database automatically or by user intervention. In one embodiment, the job database is a Virtual Job Ticket Database (VJTDB) comprising both a Virtual Print Job Ticket Database (VPJTDB) and a Virtual Finish Job Ticket Database (VFTDB) as each is defined below. At step 101, a table of printer capabilities and constraints is retrieved from the VJTDB (preferably from the VPJTDB portion of the VJTDB) and a table of finisher/assembler capabilities and constraints is retrieved from the VJTDB (preferably from the VFJTDB portion of the VJTDB). In one embodiment, the VJTDB contains data relating to all applicable devices that can be accessed although the VJTDB may be set up such that it retrieves data for only a sub-set of the total devices for any specified type of job. In addition to allowing certain equipment to be dedicated to certain types of jobs, the above arrangement enables a device that is broken, is in use, or is being serviced to be taken "off line" in relation to a job being processed when such equipment is unavailable. At step 102, the PMC uses all of the job attributes stored in the job model to map all generic combinations and sequences for

printing, assembling, and finishing that could result in creation of the finished document, i.e., B&W printing, collating, folding, gathering, etc. Each of these combinations and sequences shall be referred to as a "thread". At step 103, the PMC uses the retrieved lists of all of the capabilities and constraints of the devices described in the VJTDB to generate a list of all possible specific paths, or threads, by which the retrieved devices can implement the operations and attributes identified in the job model. In this manner, the generic threads such as shown in Figure 4 are mapped onto various paths or threads comprising specifically identified equipment and sequences. Importantly, the PMC uses the list of constraints within step 103 to determine various job segments that may be required when processing the job. For instance, if a particular thread encounters a bin height constraint of 2.2 inches from one of the devices in the thread, then the portion of the job that flows through that thread will be mapped in job segments that will all fit within the 2.2 inch bin constraint. In effect, job segments for each thread or portion thereof are defined by the combined constraints derived from every device in the thread. Where appropriate to provide additional flexibility, job segments may be broken apart into their constituent document components after leaving a device and recombined into different job segments for processing during one or more following operations. At step 104, the PMC examines the job model and each of its nodes to ensure that all document components of the job can be mapped onto specific threads that all conform to the constraints of the various equipment and can be operated without unresolvable conflicts. If every document component can be specifically mapped in such manner, then the PMC proceeds directly to step 107. If not, then, at step 105, the user is informed of the document components that cannot be specifically mapped and is given the opportunity either (1) to make additional equipment listed in the VJTDB available for this job or (2) to return to the steps shown in Figure 9 in order to modify one or more document components. At step 106, the job

model is modified in the event that document components are modified. The VJTDB list of available equipment is also modified where appropriate.

At steps 107-110, the various possible threads are matched and compared in order to eliminate or minimize equipment utilization conflicts and to determine the optimal selection of threads for implementation of the job. The optimization process can be accomplished by any number of algorithms that provide values for the various devices and evaluation criteria for determining preferences. For instance, if optimization is to be determined on the basis of the shortest time for completion of the job, then a possible algorithm for such optimization would assign a time period for each operation performed by each device on each sheet or job segment that flows through the thread. The sums of all time periods attributable to each thread would then be computed, and the combination of threads that results in the shortest production time would be selected as the optimal mapping of threads. Similarly, optimization algorithms and VJTDB data can be established for optimizing the estimated cost of a job, for optimizing or minimizing the use of a specific device, for any similar priority goal, or for any combination of optimization goals. Step 109 indicates a particular portion of the optimization process wherein the intermediate finishing capabilities of the various printers within the threads are compared and optimized. For instance, if a printer can print two separate document components in collated, non-collated, or collated set form, these three possibilities comprise three different possible threads requiring different assembler/finishing operations. These different threads may then be compared for optimization purposes. Such different threads may be created whenever printers are capable of multiple output or finishing options. At step 110, another particular aspect of the optimization process is determining which, if any, document components can usefully be combined into a single job segment during a printer and/or finishing operation and to thereby be treated as a single node having two subparts. For instance, if two

monochrome document components identified as A and B can be printed by the same printer, one thread may exist that treats these as a combined job segment that outputs the document components in collated, offset, stacked form: A,B; A,B; A,B; etc. If the subparts A and B will subsequently be separated, they may also be offset within their respective sets. Once separated, the thread may provide that document components A and B are recombined with other document segments to form new job segments applicable for the subsequent assembler/finisher operations. At the conclusion of steps 107-110, the PMC at step 111 presents to the user a recommendation for the optimal selection of threads, including information if desired of job segments, estimated time, estimated cost, etc. In one embodiment indicated by Figure 12, the threads are presented to the user in a GUI showing (1) a perspective view of a three-dimensional representation of all devices available for the job, (2) a map that shows the thread that each job segment would take through the devices, and (3) such data as the number and sequences of the various job segments required to complete the job. Such a three-dimensional representation the job flow is preferably made available to the user both during planning of the job as shown in figure 9 and as a means for tracking the job during actual production.

At steps 112-114, the user is prompted to review and approve the recommended threads prepared by the PMC. If accepted, the job goes to the integrity and tracking features of the PMC that are shown in Figure 11. If not accepted at step 13, the user in step 114 is given the option to select a map of different threads than the recommended threads, in which case the PMC is returned to step 108. Alternatively, the user may elect to design additional threads by adding devices to the list of available equipment, by modifying the constraints of available equipment (e.g. moving bin positions to change stack height constraints), by manually creating different job segments consistent with device constraints, or by similar manual override methods.

Also in this step, equipment that is unavailable due to prior use or need for service may be returned to an available state by user intervention or passage of time. The user may also use this opportunity to return to the job model processes shown in Figure 9 in order to modify one or more document components within the job (e.g., select a different lamination material in order to use a laminator device with a different temperature constraint). When the user accepts the map of threads resulting from the processes shown in Figure 10, the job is referred to the PMC processes shown in Figure 11.

Figure 11 commences with receipt of the job segmentation and map information from the processes shown in Figure 10. At step 200, an integrity descriptor is encoded for each document component within a job segment that has been determined and accepted pursuant to the processes shown in Figure 10. Such integrity descriptor contains, for each document component, such information as the number of sheets within the document component, the number of copies to be printed, and such other information that will be useful in tracking the job and in determining whether each operation has been completed with respect to each sheet of the job segment. Step 200 contains two separate paths, each comprising several sub-steps. Path A applies if the PDL file for a particular sheet or document component arrives at the PMC without a previously created integrity descriptor. At step A200A, a user is prompted to define the type of integrity descriptor applicable to each document component, including the attribute or parameters contained within such integrity descriptor. Such definition may provide the PMC with a fixed descriptor, provide for a range of descriptor values, or may direct the PMC to derive an IDC from a database on file. At step A200B, the location where the integrity descriptor will be stored is selected. At step A200C, the PMC uses the information provided by the user in step A200A and the location identified in step A200B to generate a new integrity descriptor if none is provided by the user in step A200A. At step A200D, the integrity descriptor

itself or an Integrity Descriptor Code (IDC) is added to the applicable PDL files. An IDC comprises a pointer to the location where the integrity descriptor is stored. At step A200E, the integrity descriptor or IDC is added to the appropriate node or nodes in the job database file. Preferably, as discussed above, this is a VJTDB comprising both a VPJTDB and a VFJTDB.

Alternatively, column B of step 200 describes the processes applicable when an integrity descriptor is created before the PDL files are delivered to the PMC. At step B200A, the integrity descriptor is read from the PDL file. At step B200B, the integrity descriptor is encoded into the VJTDB or other database.

At step 201, the PMC uses the job model to determine if the document component comprises preprinted or non-printed sheets or other items that must be added to the workflow from inventory in order to complete the job. Among the items that may be noted are separator sheets, binder materials, preprinted pictorial inserts, perfume-containing scent cards, and similar items that are not printed as part of the current job. Since stacks of such inventory materials typically do not have a traveler sheet prepared for this job, the PMC at step 201 generates a Fetch Sheet PDL and directs its printing if necessary.

At step 202, the Job Segment Identifier Code (JSIC) for the job segment is generated. The JSIC is a unique identifier code that enables the FMC in Phase 3 to lookup a description of the job segment within the VFJTDB. A JSIC may be generated by combining a job number with a sequentially generated number that represents the stack number within the job. More details concerning the JSIC and its use are provided below. At step 203, the PMC directs the generation of a PDL file for printing of a Job Segment Identification Sheet (JSIS) if required. The JSIS will contain the JSIC, a human readable instruction sheet for processing of the job taken from the job model, and lists of document and document component attributes. It

should be noted, however, that in systems where the identity of a job segment can be determined or tracked without the printing of a JSIS, then some or all of step 203 may be eliminated. For instance, where the contents of the top sheet of a stack can be automatically read by a VJTR and matched against a database of expected top sheets, the FMC then could identify and track the job segment by reading such top sheet as it moves through the various production phases without the need for a special JSIS or even special JSIC on the top sheet, i.e., the contents of the top sheet itself would form a JSI.

At step 204, the PMC stores the JSIC, the job segmentation, and the selected job segment thread information in the VJTDB. At step 205, the PDL files covering a JSIS, if any, is added to the PDLs and other files required for printing of the applicable job segment and such files are delivered to such printer for printing. Phase 2 of the printing production process is thereby commenced. At step 206, fetch sheets for the job are also sent to appropriate printers in order that they be available to the human operators as the job is produced. At step 207, the PMC determines whether the job has additional job segments to process. If yes, then steps 200-206 are repeated again for each job segment. Once processing of all job segments has been completed, then at step 208, the PMC directs creation of a traveler sheet PDL and its printing in order that all of the information required to produce and process the job and each of its job segments and/or document components is placed in human readable form for the human operators. Such traveler sheets typically stay with the job jacket as it moves around the print shop.

More details concerning the VJTDB will now be discussed in relation to Figure 13. In particular, the VFJTDB portion of the VJTDB will be described in more detail. As described above, the VFJTDB is a database or a data file that contains all job construction, control and integrity data necessary to take the prints coming from the printing device(s) and perform the necessary finishing processes to turn the prints into the desired final

document form. As such, one portion of the VFJTDB or other accessible database contains tables or lists of all devices accessible for operations occurring printing of sheets has occurred, including, without limitation, the availability and priority parameters for each device and lists of capabilities and constraints for each device. Another portion of the VFJTDB contains data pertaining to each particular job.

The job description portion of the VFJTDB is best understood as a database having a hierarchical tree structure with each level of the tree having one or more discrete nodes. Figure 13 shows such a tree structure as it may apply to production for a signature document. The top level Sig1 represents the basic job description. Encoded within this node are such details as job name and number, the node identifier (in this case, Sig1). information concerning the client, accounting and billing information, scheduling data such as date of order and scheduled production and delivery dates, etc. These and other details are the "attributes" associated with node Sig1 and may be stored in table form within the node itself or may be associated by reference pointers within Sig1 to external databases. Also within node Sig1 is typically a reference pointer to any "parent" nodes and to the nodes in the level immediately below Sig1. Such reference pointers to nodes immediately above and below a particular node is a characteristic at all levels of this embodiment of the VFJTDB. By following the trail of such pointers, an entire job can be reassembled if such one node is identified.

On the next level below Sig1 is node P1 which is the node containing descriptors for a particular document to be produced as part of Job Sig1. There is no limit to the number of documents that may be stored under a particular job, and nodes P2, P3, etc. could exist. Turning now to node P1, such top level document node contains an identifier of the document form governing the document. By designating such document form, a table of attributes for such document form and a set of Document Form Rules are

associated with P1, and these rules and attributes will be used by the PMC when mapping production of the job. As noted above, node P1 also identifies each of its children nodes. An important aspect is that each node P1 contains information detailing the order and manner in which each of the children nodes relate to each other, e.g. cover C1 is identified as the document component that will ultimately be on the outside of the document. Since the example shown in Figure 13 is a signature booklet, the relationship between the children of P1 can be specified by the order in which each document component is added to the gathered signature stack prior to its folding and/or binding, as the case may be. Also contained or referenced in node P1 is a job model register in which the PMC may store data describing threads relating to production of the job. When filled, such register will ultimately contain such detailed information as a "Build Sequence" for document P1. An example of a Build Sequence for the shown job might be:

1. Feed 1 sheet of job segment C1 from Bin #1 of Printer #4 and place the sheet in Bin # 6 of Assembler/Finisher #2.
2. Feed a first offset set (comprising one document component within a job segment stack) from job segment B1 located at Bin #1 of Set Feeder #1 and place the set on top in Bin #6 of A/F #2.
3. Feed job segment I1 (which comprises an insert) from Bin #20 of inventory delivery system #1, re-orient the set by a 90 degree rotation, and place insert I1 on top in Bin #6 of A/F #2.
4. Feed a second offset set (comprising a second document component within a job segment stack) from job segment B1 and place the second set on top in Bin #6 of A/F #2.
5. Feed job segment I2 (which comprises a second insert) from Bin #5 of inventory delivery system #1 and place insert I2 on top in bin #6 of A/F #2.

The job model register will also contain information, conforming to the applicable Document Form Rules, for programming A/F #6 to perform all of the scoring, folding, stitching, trimming, etc. operations that will result in final production of the signature booklet described in node P1. Among the typical parameters encoded for such operations will be the identity of the operation and the location for its operation on the work piece, the type and temperature of any lamination step, any pressure requirements for scoring, folding, binding, etc., the type and location of binding, etc.

Each node at the level below node P1 will contain information relevant to the job segment represented by such node. Such information will include the printer and bins from which such segment is derived, the number of sets or sheets in the job segment, its current location, attributes of the segment such as whether it is collated or not collated, face up or face down, orientation, sheet weight, color, type and thickness, etc. Of particular note in the shown example is node B1 which comprises a single job segment of two collated document components arranged in interleaved offset sets. Each of these document components may be described in their own separate nodes as indicated by node S1. There is no limit to the number of nodes or levels of nodes in the present invention. In the event that each of the document components represented in B1 were sourced from separate printers, then each may would comprise a separate job segment with a separate node which then became combined into a new job segment at node B1. Note that when delivered from node B1, the job segment stack was disassembled and its components were rearranged into a new job segment P1. Obviously, for complex books comprising many signatures, inserts, covers, etc. many nodes and levels of nodes would be represented in the VFJTDB.

Turning now to Figure 14, more details are provided concerning the Finishing Module Controller (FMC) shown as box 700 in Figure 5. Beginning at step 600 of Figure 14, two methods of entering the JSIC for each

job segment is shown. In step 600A, the VFJTR reads the JSIS or other JSI from a job segment of the job. As described above, the VFJTR may read a bar coded JSIC, a glyph code, optical characters with an OCR, magnetic or optical memory, or any other encoding or symbology which a machine is capable of interpreting. Alternatively to step 600A, step 600B indicates that the JSIC could be read and entered by a human operator. At step 601, the FMC receives the JSIC data from the VFJTR or human operator. At step 602, the FMC queries whether the data received comprises all of the job model data for the job. As described above, a complete set of job-related data could be encoded by a glyph or other method on each JSIS of the job. Region 602 of the VFJT shown in Figure 7 contained such information. If the FMC has not received a complete job model from the operator or from the VFJTR, then the FMC at step 603 makes a call upon the VFJTDB for a complete set of the relevant job model data. In order to do so, the FMC communicates the relevant JSIC to the controller for the VFJTDB (which may be the PMC or the FMC or other controller) in order that the VFJTDB may locate the identified job segment node. Using data recorded within this node, as described above in relation to the VFJTDB, a complete job model and information from every node of the job can be extracted from the VFJTDB. At the conclusion of step 603, processing returns to step 602 where the FMC again inquires whether it has received a complete set of job model data. Once this query is satisfied, then, at step 604, the FMC reads the job model data to identify all job segments of the job. Since even job segments comprising identical sheets have unique JSICs, a complete list of all JSICs provides the FMC with the complete job flow, i.e., not only a description of how to build a single finished product but also a list of all of the job segments that will "flow" through the assembler/finisher process to build all of the products. See also Figure 12 for an indicator of the data available to track both individual product units and the entire production process for all sub-units. At step 605, the FMC extracts from

the VFJTDB or other database data concerning the location and status of all of the job segments identified for the job. In a continuous printing operation, some of the job segments may yet wait processing1 within the printers, and this status is communicated to the FMC. At step 606, the FMC determines, based upon the job model data, whether all or a sufficient quantity of the job segments are ready and available for the assembler/finisher operations. If not, then at step 607 the FMC notifies the human operator and waits for further instruction. If yes, then the FMC has determined that the work pieces are ready for assembling and finishing operations.

The FMC begins its investigation of equipment readiness at step 608. At step 608, the FMC identifies from the job model each device necessary for the next set of finishing operations. In most cases, assembler/finishing operations will be designed to produce finished products in a continuous assembler/finisher process. Where the assembler/finisher process may conveniently be broken into non-continuous phases, however, the FMC need only identify the assembler/finisher devices necessary for the next phase of operations. Also as part of step 608, the FMC extracts from the job model any device configuration attributes that the job model requires in order for the job to be implemented. For example, if an assembler device permits different input bin configurations and the job model specifies a specific configuration, then this data would be extracted from the job model. Similarly, if paper path guides or scoring and folding apparatus, etc. need to be in specified locations that are not adjustable through automatic programming, such attributes are identified through the job model. At step 609, the FMC interrogates the listed devices to determine whether they and the specified configurations are available for processing of the job. The interface between the FMC and the assembler/finisher devices may take many forms. One such interface protocol is the Modular Feeding and Finishing Architecture (MFFA) of the Xerox Corporation. The MFFA is described in a number of U.S.

patents, including US-A-5,701,557 issued to Webster. If the devices and configurations are not currently available, the FMC inquires, at step 610, whether the MFFA or other interface protocol combined with the programmability of the device enables the FMC to program the applicable devices to make each device and its specified configuration available. If yes, then the FMC returns to step 609. If not, then at step 611, the user is notified. If the user can and chooses to place the devices in the desired status, then step 612 indicates that the user implements these changes. Once implemented, the user signals that the changes have been made, and the process returns to step 609. If the user cannot or elects not to make the required changes, then the job is paused unless the user elects to create a different job model using different threads that avoid the constrained condition. To do so, at step 613, the user returns the process to step 105 shown on Figure 10 where new threads are created. New job segments may also be recommended since a different set of constraints may be encountered. As part of creating new threads, the unavailable device or configuration would be removed from the VFJTDB list of available equipment. It is important to note that although Figure 15 indicates that the process is returned to the PMC for this task, the task of could also be completed within the FMC. As indicated in Figure 5, whichever of the PMC or the FMC implement this task, the results are communicated and stored in the job model portion of the VFJTDB and are available to the other controller.

Once it is determined, at step 609, that the required devices and configurations are available for the job model first selected or for a subsequent revised job model, then the FMC proceeds, at step 614, to specifically define where and how the various job segments are to be loaded into each of the devices selected for use. Some of this information may have been determined by the PMC processes described in relation to Figures 8-11. However, details such as bin selection and specific orientation (face up, face

down; binding edge first or last, etc.) may not have been selected during Phase 1 of the job in order that the greatest flexibility be preserved until the FMC is invoked to implement specific threads and configurations for assembler/finisher operations. At step 615, the FMC verifies, through communication with the assembler/finisher devices, all bins that should be loaded are in fact loaded properly. Instructions are reissued if needed. Assuming that all bins have been properly loaded, then, at step 616, the FMC automatically programs devices that are automatically programmable to operate in accordance with the job model. Programming instructions may include, where appropriate, parameters computed by the FMC itself from capability and constraint data extracted from the VFJTDB. For instance, the FMC may determine feed rates for a device based upon the constraints of another device operating in conjunction with the first device. For those devices that are not automatically programmable by the FMC, the FMC instructs the operator concerning such programming. Such instruction set may take the form of a printed JSIS. Once proper programming is verified at step 617, the set-up portion of the FMC function is over.

Beginning at step 618, the control, tracking and integrity functions of the FMC begin. At 618, the FMC issues a Run Command to start the assembler/finisher processes. Alternatively, the FMC issues a Run Release signal to the operator that notifies the operator the assembler/finisher devices are in a ready condition. Once operations are initiated, the FMC, at step 619, monitors and tracks performance of the job and at step 620, issues appropriate control commands in response to tracking and performance data. Part of the tracking function uses the sheet counting capabilities of various devices to count sheets. The FMC similarly may track each job segment by its JSI as it moves through the process. In so doing, the FMC may issue calls for job segments to be loaded into devices in sequential order, e.g., issue orders to refill bins even before "bin empty" signals are generated or orders to

empty bins filled with completed products or job segments. The FMC may also be in contact with performance of each device and may issue commands to adjust feed rates in order to maintain feed sequences within optimal ranges. The FMC may also monitor consumption of supplies such as binding materials and issue calls for reloading of such supplies when appropriate.

Integrity and control functions are indicated at steps 21- 62__. At step 621, the assembler/finishers are programmed to send to the FMC jam or job stop notices. The FMC then coordinates appropriate pauses in other devices dependent upon the jammed or stopped device. At step 622, the FMC interrogates the stopped or jammed device for error analyses. At 623, the FMC issues instructions to the operator. These instructions may as simple as informing the operator which device is the cause of the problem or might be as complex as providing full recovery instructions. At 624, the FMC verifies whether the jam or stop conditions have been removed. If not, then the sequence returns to step 622. If the conditions have been removed, then the FMC at 625 issues restart commands which may include reloading and reconfiguration commands to the operator and/or devices. Once restarted, the FMC returns to step 621 where the process continues.

At step 626, the FMC maintains integrity data. Based upon count data and job segment status data obtained in steps 619 and 620 plus the stop and jam recovery data from steps 621-625, the FMC tracks which sheets have been lost, destroyed, or are unaccounted for during the assembler/finisher process. The FMC also tracks which job segments or finished products may contain missing sheets based upon the counting and tracking data. Based upon this integrity data, at step 627, the FMC issues instructions for additional printing and/or assembler/finisher operations. These instructions could be automatic to the appropriate devices or may be instructions to the operator regarding manual reprogramming of certain portions of the job. One advantage of having an FMC that is capable of

tracking even complex jobs through all parts of the assembler/finisher operation is that there is less incentive to print extra copies of each document component, thereby saving printing and inventory cost. The lowered incentives result because the FMC's ability to track job segments and sheets within job segments enables operators to more precisely know when and where defects have occurred.

At step 628, the FMC sends its tracking and integrity data to a central database such as the VFJTDB in order that a record of the job performance be kept. This step 628 will occur upon completion of the production run but could occur at any interim step. The PMC or other controller can use this data to account for the number of completed finished units, the amount of wasted sheets, time for production, and any other criteria that is desired to be monitored and recorded. Lastly, at step 629, the FMC automatically updates the VFJTDB concerning any new capability and constrain data relating to the devices. Such data may include reliability data, out-of-service data, new information regarding feeding constraints, etc. In sum, by breaking an assembler/finisher job into trackable job segments, the FMC enables the systematic control and integrity monitoring of offline assembler/finisher devices that until now have only minimally been digitally connected to each other or to devices such as printers.

An embodiment of key portions of software for the present invention will now be described. As described above in relation to the VFJTDB, each job may be comprised of an indeterminate number of document components and/or job segments. This indeterminate feature necessitates software algorithms that do not use predefined limits and boundaries, such as static memory buffers. Instead, the software algorithms recursively search the job model in the VFJTDB or other database based on a job key sequence. The job key sequence is provided from a data base record related to an individual job segment. By using a Microsoft Visual Basic 6.0

Predefined TreeView Component, the "Node Key" is constructed and stored using the job key sequence. This allows the user to select a node on the graphical display in the form of an icon and for the job sequence key to be retrieved in the form of the "Node Key". The data base job model can then be recursively searched to facilitate processing requirements.

The following software demonstrates the recursive algorithm used to build the graphical TreeView representation of the physical product being built. The Node Key is constructed as the sequence of parent node keys. Any time after the node icon is selected, the Node Key can be accessed and used to efficiently locate the Node in the over all job model. This subroutine builds the tree from the root down by recursively calling itself and stops at each branch of the tree when no more children exist. The advantages of using this algorithm are:

1. No predefined limits are defined for the number of levels existing in the job model tree;

2. The code size is compact;

3. The code uses existing built-in Microsoft Visual Basic Controls to provide the graphical presentation; and

4. The code retrieves the sequence key efficiently when processing a user selection event.

This software code is as follows:

```
Private Sub FillTree(Childid As String, Key As String, _
                nodeindex As Integer, nodx As Node, keys() As String, _
                level As Integer, nf As Form)
'a recursive building of the job node tree
Dim pd As NodeDef
Dim pdkey As String

NumChildren = GetChildren(Childid, pd)
If (NumChildren > 0) Then
 For x = 1 To NumChildren
```

```
    pdkey = Key & "," & pd.NodeIds(x) & "," & Str$(nodeindex)
    nodeindex = nodeindex + 1
    Set nodx = nf.TreeView1.Nodes.Add(Key, tvwChild, pdkey, _
                        pd.NodeIds(x) & " " & pd.Dscr(x))
    'Seems senseless but this makes entire Tree Visible
    nodx.Selected = True
    nodx.Selected = False
    If (keys(level) = pd.NodeIds(x)) Then nodx.Checked = True
    Call FillTree(pd.ChildIds(x), pdkey, nodeindex, nodx, keys, level + 1, nf)
  Next x
End If
End Sub
```

Another algorithm of the FMC portion of the present invention presents a subroutine that recursively calls itself to check that all Nodes in the tree are in a required state. Every Node in the entire job model tree is checked until one is found not in the state. In this case, the finishing equipment is being checked to see if each bin is in the LOADED state. The function CheckSegments returns a TRUE or FALSE value. The actual purpose of this subroutine can be easily modified by simple substituting a new function in place of CheckSegments. This allows this code to be cloned and used for a number of purposes in the FMC.

The advantages of using this algorithm are:

1. No predefined limits are defined for the number of levels existing in the job model tree;
2. The code size is compact; and
3. The code is easy to clone and reuse for a number of FMC and related tasks.

An example of the software follows:

```
Private Sub JobRunTree(Childid As String, Abort As Boolean, _
                       FormIdx As Integer, _
                       JobIdStr As String, _
                       nf As Treeview, _
                       NodeIdx As Variant)
'recursive traverse of the job node tree used buy Sub JobRun
Dim pd As NodeDef
```

```
Dim NumChildren As Integer

If (Abort = False) Then
 NumChildren = GetChildren(Childid, pd)
 If (CheckSegments(FormIdx, JobIdStr, NumChildren, _
      nf.Nodes(NodeIdx).Key, NodeIdx)) Then
  Abort = True
 Else
  If (NumChildren > 0) Then
   For x = 1 To NumChildren
    Call JobRunTree(pd.ChildIds(x), Abort, FormIdx, JobIdStr, nf, NodeIdx)
   Next x
  End If
 End If
End If


End Sub
```

Another aspect of the present invention is a software user interface and a "Document Construction Wizard" (DCW) software to assist the user in linking electronic image data files to particular document components and then to organize document components into documents. A user creates electronic image data files that represent the images that make up the document(s) that is desired to produce. The user chooses the final document form from one of the listed document form types. He/she then designates each image data file(s) as a specific document components of one of the seven document forms. The software enables and the wizard assists the user in designating document component orientations and orders within the document form. It also uses a different set of rules for each document form to limit the number and type of document components from which a user can construct the document. For example, a signature booklet cannot have two separate sheets of media that are covers. It can only have one cover (a wrap around cover) but a tape bound document form may have up to two covers (front and back). It prompts the user for the required attributes that are

dependant upon the selected document component type or and document form. This software also reads a file or database that represents capabilities and constraints of the finishing equipment available to the user and thus would only display the finishing operations, options or attributes that are available based upon that finishing equipment that is available.

The DCW would also prompt the user to enter whether or not the component is Variable Component. A Variable Component is a component whose image data will change from one copy of the document to the next copy of the document in the print job. A Static Component one whose image data will not change from one copy of the document to the next copy of the document in the print job. If the component is designated as Variable Component the user would have to designate a form file name and a data file name. All of this data would be stored in a database.

The DCW would allow the user to designate whether or not any document components are an External Component or whether it is to be generated from an image data file. A External Component is one that is produced or printed through some other process but is not to be printed as part of this job. If designated as an External Component then the DCW would prompt the user for an integrity descriptor for the preprinted document component. The integrity descriptor could be entered or read from a file if the user enters the integrity descriptor file name. All of this data would be stored in a database.

Turning now to Figures 17-19, another aspect of the present invention comprising a Graphic User Interface (GUI) will be described. This GUI is designed to enable facilitate operators to load and operate assembler/finisher devices even when the operators do not have expert knowledge of the how the individual product components are to be combined to form the final document product. In the prior art, it has often been necessary for the creator of the job to either directly assist a machine

operator, to operate the devices directly, or to write a detailed set of process instructions.

The GUI of the present invention provides detailed information required by an operator who has no previous knowledge of the document to be constructed by the finishing equipment. The interface provides information ranging from a simple job overview to very detailed component status information. Information is presented to the operator describing the product and each of its components. The operator is also guided through how to load and operate a variety of finishing equipment. The GUI also permits the operator to interact with a VFJTDB or other database. The GUI is designed to enable the user located at the finishing equipment location to both run and load the finishing equipment and also to stage jobs. This user interface also be used to facilitate a materials packaging task at the output location of the finishing device(s) or printers. For example: Machine operators could use this GUI to assist in palettization of boxes of output being shipped to other locations. The GUI can also be used at the printer output location for the purpose of packaging segments to be transported to a remote finishing location.

.Turning now to Figure 17, an embodiment of the initial GUI screen is shown. Using this screen, an operator can select the database and job name. This screen also enables the user to add, remove or similarly manipulate the document component nodes in the database.

Turning now to Figures 18 and 19, two embodiments of a Job Assembly View of the present invention are shown. In both embodiments, the Job Assembly View provides a graphical view of the job model arranged as a hierarchical tree. Microsoft Visual Basic 6.0 Predefined TreeView Component can be used to create this hierarchical tree arrangement. The Job Assembly View can use text descriptions as well as graphic icons. For example, a

picture of a signature book or a picture of a cover to a book could be displayed as well as an actual bitmap photographic image of the product.

Figure 18 shows an embodiment of the Job Assembly View that references the job from the unique job identifier code described above. Figure 19 shows an embodiment of the Job Assembly View that references the job from a combo box status list of jobs.

Use of the GUI of the present invention is typified as follows: First, a user scans a Job Segment Identifier Sheet (JSIS) or otherwise enables the system to determine the JSI for a job segment using the procedures described above. As described above, the PMC, working in conjunction with the VFJTDB or other database, is able to reassemble the job model and all of its nodes from a single JSI. Once the job and all of its nodes have been identified, the PMC, FMC, or other terminal can create the Job Assembly View for the job if the user selects "Segment Notification".

Second, the user can view a status list of jobs in a standard combo box and pick which job is desired to view.

Third, the user will be able to edit the Job Assembly View graphical tree by using a tool box drag and drop action. For example the user may add or take away components from the product.

Fourth, an unlimited number of Job Assembly View windows can be displayed. This feature will be useful for staging jobs for latter processing.

Fifth, the main form of the GUI will display the job status, which is the printer and/or assembler/feeder device queue status.

Sixth, segment identifiers such as JSICs may be displayed and edited by using the keyboard.

Seventh, when a user requests that a job be run, the System software (which may reside in the PMC, FMC, both or other clients) will diret the GUI to display a status check of each job segment and will notify the user

should there be missing or incomplete job segments. This feature of the GUI was described above in relation to the FMC.

Eighth, a system diagnostics form will be enabled and accessed from the main form.

Ninth, the type and level of information for every node or level of the job model may be selected by the user, including, without limitation, information relating to the job node, document node, document component nodes, job segment nodes, etc.

Tenth, the product tree representation shown by the Job Assembly View of the present invention is unlimited in size and the number of levels . For example, the highest level could be a pallet of boxes filled with a variety of documents, books , and other objects (such as CD ROM disks) while the same Job Assembly View tree could have a lowest level comprising a single page of a signature book. As described above in relation to the PMC, specific PDLs will be associated with such sheet and can be identified and extracted from the Job Assembly View tree GUI.

In sum, what has been presented is a system for electronic management and control of a wide range of finishing processes characterized by input from multiple production operations and equipment that, depending upon the job, may be variably applied to work pieces that themselves are highly variable between different jobs. The present invention has been demonstrated in relation to printing and finishing operations for printed documents. The principles of the present invention, however, apply to such production and finishing systems as, without limitation, textile production (which may include printing, cutting, sewing, and finishing), packaging operations for various consumer and industrial products, and printed wiring board production, etc. The present invention is particularly applicable to many operations where processes for production of work pieces are managed separately from processes for finishing and packaging of such work pieces.

Among the advantages demonstrated are:

1) an integrated digital architecture for interactive control, tracking and integrity functions of all three phases of the prepress/printing/finishing process;

2) an integrated apparatus and method for enabling an operator, prior to printing of a job, to provide complete instructions for complex assembler/finishing operations, especially if off-line from the printer controller;

3) a method for accurately and completely describing the final document form of a complex document in a manner that enables both printer controllers and assembler/finisher controllers to instruct and control their respective printing and assembling/finishing operations;

4) an apparatus and method for dividing and managing a print job and associated work flow in response to constraints of both the available printing systems and the available finishing systems;

5) a queue management system that manages the entire print and finishing process in accordance with the availability of specified printers and of specified assembler/finisher equipment;

6) an interactive integrity check system capable of tracking each sheet through each production process, especially those portions of the process involving assembler/finisher operations that are offline from the printer controller; and

7) apparatus and method, acting in response to an interactive integrity check function, that automatically instructs printers to add or replace sheets not available to the assembler/finisher apparatus when such assembler/finisher apparatus calls for such sheets.

It is, therefore, evident that the present invention fully satisfies the aims and advantages set forth above. While the invention has been described in conjunction with several embodiments, it is evident that many alternatives, modifications, and variations will be apparent to those skilled in the art. Accordingly, it is intended to embrace all such alternatives, modifications, and variations as fall within the spirit and broad scope of the appended claims.

WHAT IS CLAIMED IS:

1) A printing and finishing system for printing and finishing work pieces of a job, comprising:

a) a printing device for producing the work pieces of the job;

b) a finishing device for finishing the output of the printing device, such finishing device being controlled separately from the printing device and having at least one constraint;

c) a production monitor controller that outputs job coordination information, which coordination information is based at least in part upon constraints of the finishing device; and

d) a finishing module coordinator that, after receiving job coordination information output from the production monitor controller, directs operation of the finishing device.

2) The production and finishing system of **claim 1**, wherein the production monitor controller outputs job coordination information comprising identity of job segments determined at least in part upon constraints of the finishing device.

3) The production and finishing system of **claim 2**, wherein the production monitor controller outputs at least a portion of finishing job segment information prior to production of at least a portion of the job by the production device.

4) A system for integrating and controlling finishing processes, comprising:

(a) a production monitor controller capable of separating a production job into job segments based upon the capabilities and constraints of devices to be used in the production process;

(b) at least one database for storing information concerning the capabilities and constraints of devices to be used in the production process and for storing job segment descriptions;

(c) a finishing module coordinator, in communication with assembler/finisher devices and with at least one database, for tracking job segments during the production process.

5) A method for coordinating the printing and finishing of a print job, comprising:

a) printing job segments using a printing device having at least one constraint;

b) finishing the printed job segments using a finishing device that is controlled separately from the printing device and having at least one constraint;

c) outputting job coordination information from a production monitor controller, such job coordination information being based at least in part upon the constraints of the finishing device; and

d) directing operation of the finishing device by a finishing module coordinator after such finishing module coordinator receives job coordination information from the production monitor controller.

6) In a finishing system having at least one finishing device that is controlled separately from production equipment and having a controller with access to device-dependent parameter information, including constraints for each finishing device, with access to a description of the workpieces of a job,

and with access to the manner in which such workpieces are to be finished, a method for an automated production monitor control function, comprising:

(a) selecting at least one finishing device for use within the finishing system;

(b) identifying constraints of the selected device; and

(d) specifying, with the controller, segmentation of the workpieces such that the attributes of each segment do not exceed the set of identified constraints.

7) The method of claim 6, wherein:

(a) the step of selecting further comprises selecting at least two devices for performing the finishing operation in sequential order; and

(b) the step of identifying further comprises creating a set of combined device-dependent constraint parameters relating to the combination of selected devices.

8) In a finishing system having at least one database for storing information concerning the capability and constraint attributes of devices to be used within the system and for storing job segment description information and having a description of the components of a job together with the order in which the components are to be assembled, a method for a production monitor controller, comprising:

(a) retrieving from the at least one database information concerning the capabilities and constraints of devices to be used within the assembler/finisher system;

(b) selecting at least one device within the assembler/finisher system for processing of the job;

(c) determining the combined constraint attributes of the selected at least one device; and

(d) segmenting the workpieces of the job such that the attributes of each segment do not exceed the combined constraint attributes of the selected at least one device.

9) In a finisher system having at least one database for storing information concerning the capability and constraint attributes of devices within the system and for storing job segment description information and for storing a job model that includes a description of the components of a job together with the order in which the components are to be assembled, a method for a finishing module coordinator, comprising:

(a) retrieving job segment and job model information from the at least one database;

(b) determining the status of job segments;

(c) determining the status of devices to be used for processing the job; and

(d) monitoring performance of the job as the devices operate.

10) In a finishing system having finishing devices with capability and constraint attributes and having a production monitor controller that determines job segments of the job based upon the capability and constraint attributes and that determines an hierarchical description of the job and its components, a method for a database system, comprising:

a) storing capability and constraint attributes in the database;

b) communicating the capability and constraint attributes to the production monitor controller;

c) creating a job model location within the database for storing a description of the job and its components, including job segments;

d) receiving from the production monitor controller information that describes the job and its components, including descriptions of job segments of the job; and

e) storing the description of the job and its components, including job segments, in the job model location within the database.

11) In an finishing system having at least one database for storing information concerning the job model for a job, including identification and descriptions of the job and job segments stored as nodes of information within the database, a graphical user interface, comprising:

a) a opening screen with user options to select a database, a job stored in the database, and a type of report for displaying information concerning the job;

b) an optional display of a selected job in a hierarchical view with multiple nodes arranged in a tree view showing the relationship of each node to other nodes within the same job; and

c) an optional display of information describing the status of each job segment within the job.

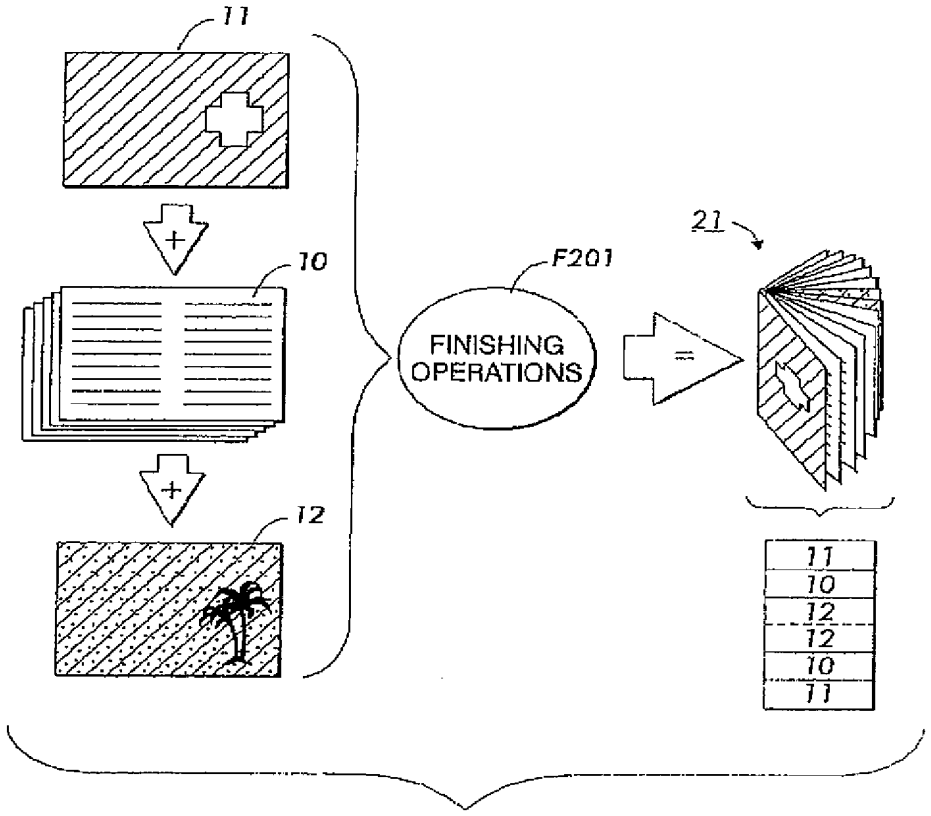


FIG. 1

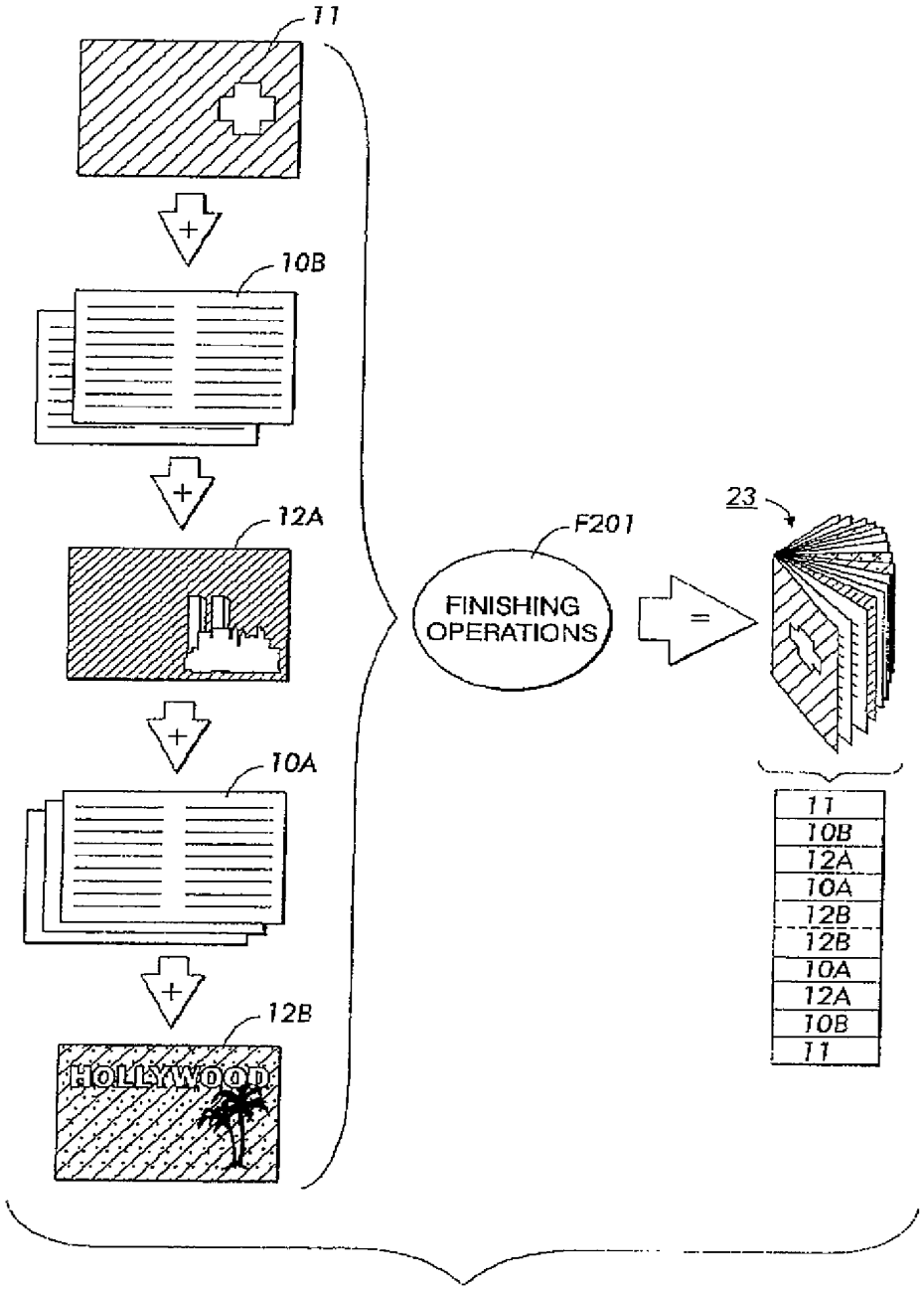

11
10B
12A
10A
12B
HOLLYWOOD
F201
FINISHING OPERATIONS
23
11
10B
12A
10A
12B
12B
10A
12A
10B
11

FIG. 2

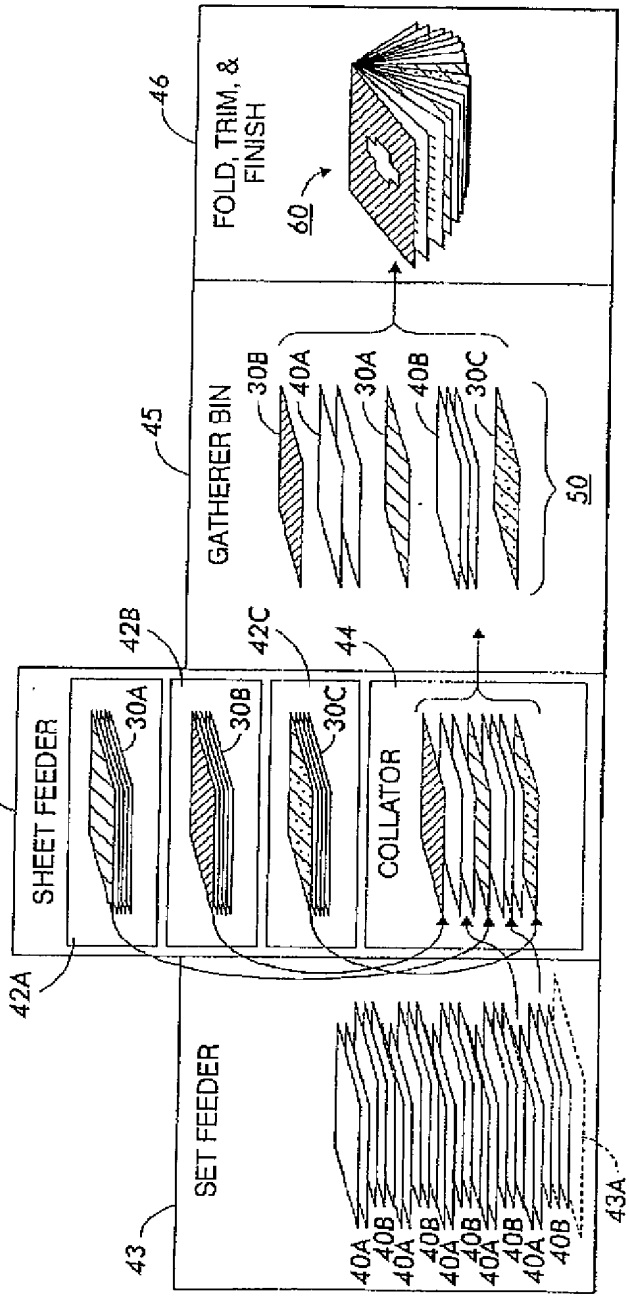

42
SHEET FEEDER
30A
30B
30C
COLLATOR
42A
42B
42C
44
45
GATHERER BIN
30B
40A
30A
40B
30C
50
46
FOLD, TRIM, & FINISH
60
43
SET FEEDER
40A
40B
43A

FIG. 3

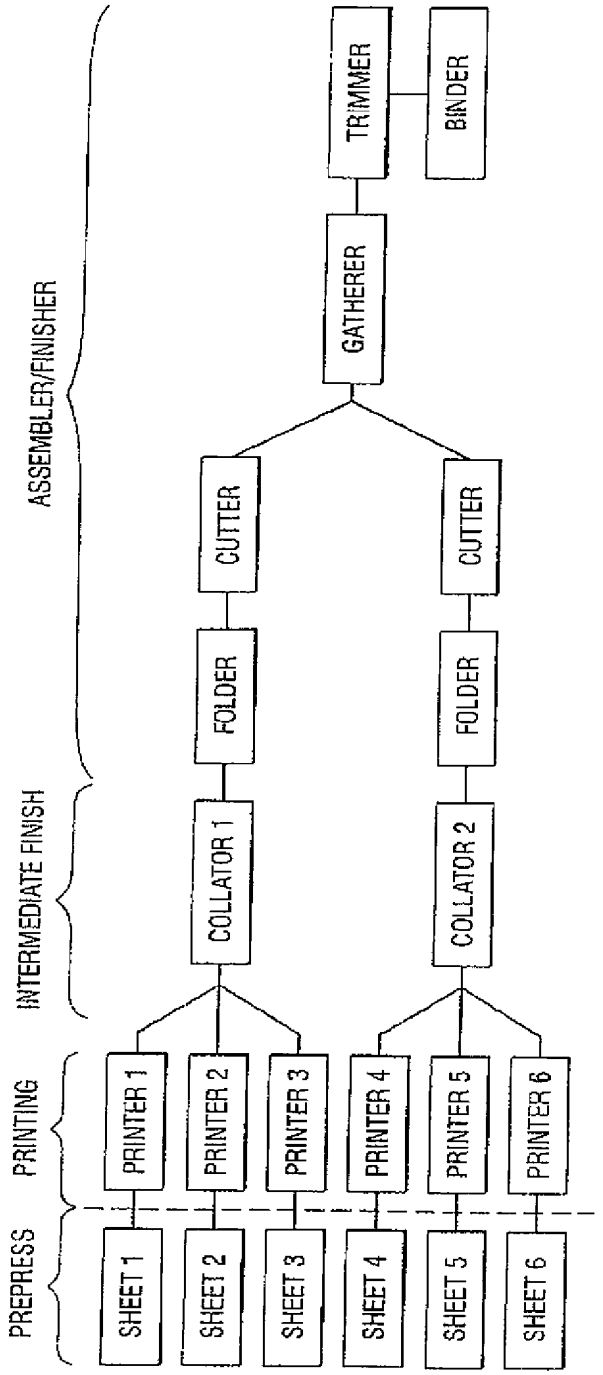

PREPRESS
PRINTING
INTERMEDIATE FINISH
ASSEMBLER/FINISHER
SHEET 1
SHEET 2
SHEET 3
SHEET 4
SHEET 5
SHEET 6
PRINTER 1
PRINTER 2
PRINTER 3
PRINTER 4
PRINTER 5
PRINTER 6
COLLATOR 1
COLLATOR 2
FOLDER
FOLDER
CUTTER
CUTTER
GATHERER
TRIMMER
BINDER

FIG. 4

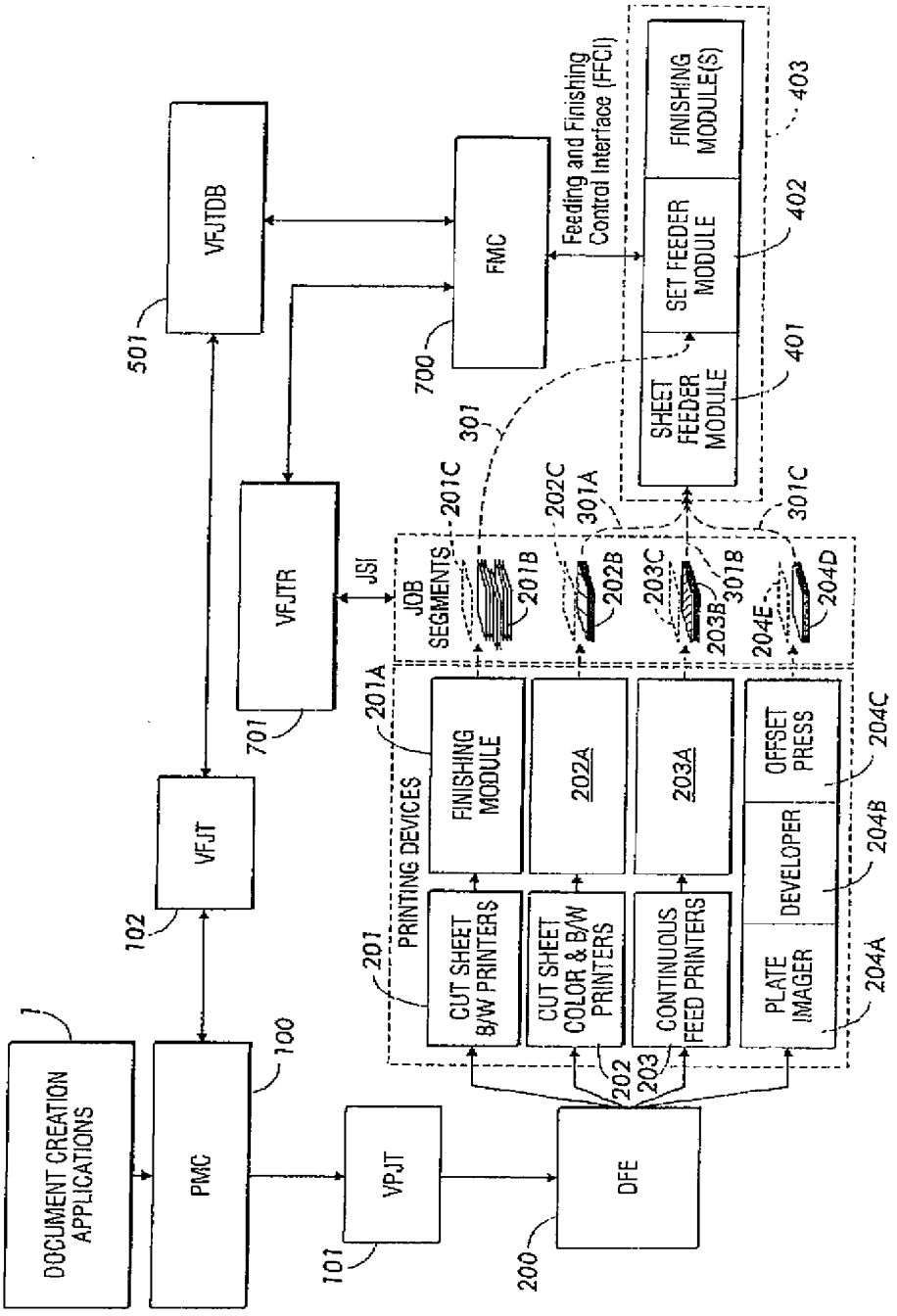

DOCUMENT CREATION APPLICATIONS
1
PMC
100
VFJT
102
VPJT
101
DFE
200
VFJTR
701
JSI
VFJTDB
501
FMC
700
Feeding and Finishing Control Interface (FFCI)
403
FINISHING MODULE(S)
SET FEEDER MODULE
402
SHEET FEEDER MODULE
401
301
JDB SEGMENTS
201C
202C
301A
301C
201B
202B
203C
203B
301B
204E
204D
PRINTING DEVICES
201A
FINISHING MODULE
202A
203A
OFFSET PRESS
204C
DEVELOPER
204B
PLATE IMAGER
204A
CUT SHEET B/W PRINTERS
CUT SHEET COLOR & B/W PRINTERS
CONTINUOUS FEED PRINTERS
201
202
203

FIG. 5

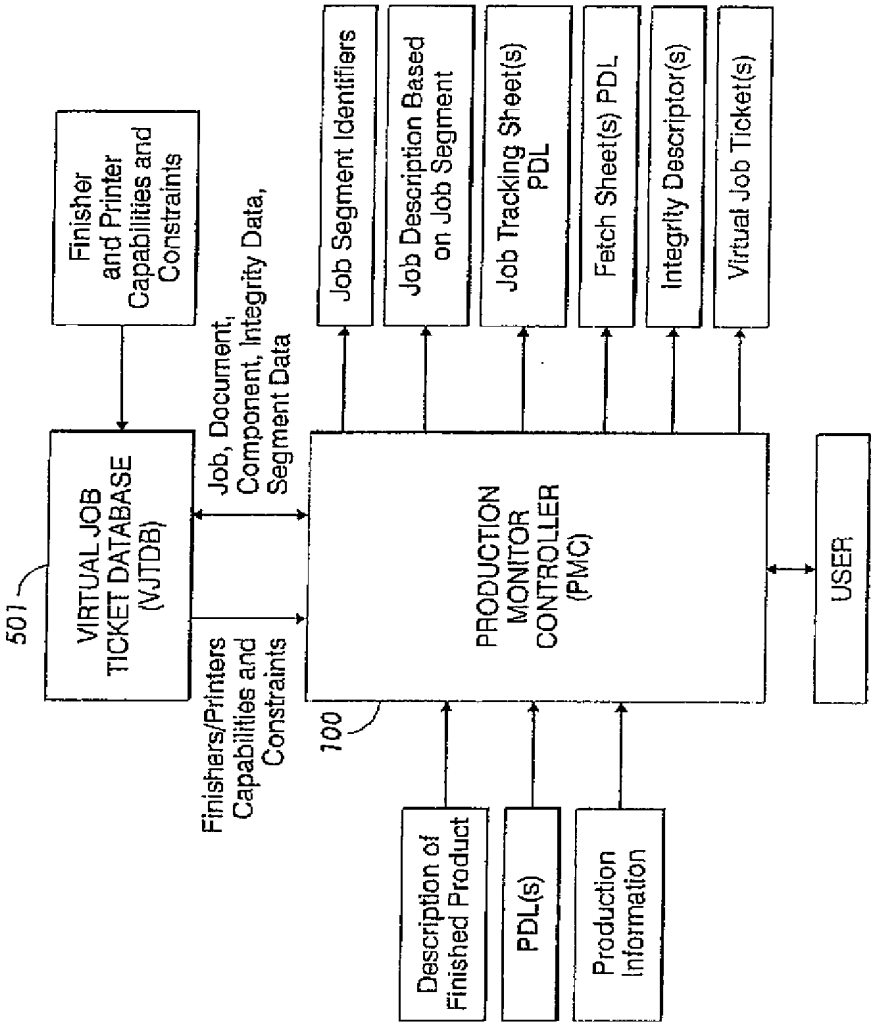

Finisher and Printer Capabilities and Constraints
VIRTUAL JOB TICKET DATABASE (VJTDB)
501
Job, Document, Component, Integrity Data, Segment Data
Finishers/Printers Capabilities and Constraints
PRODUCTION MONITOR CONTROLLER (PMC)
100
USER
Job Segment Identifiers
Job Description Based on Job Segment
Job Tracking Sheet(s) PDL
Fetch Sheet(s) PDL
Integrity Descriptor(s)
Virtual Job Ticket(s)
Description of Finished Product
PDL(s)
Production Information

FIG. 6

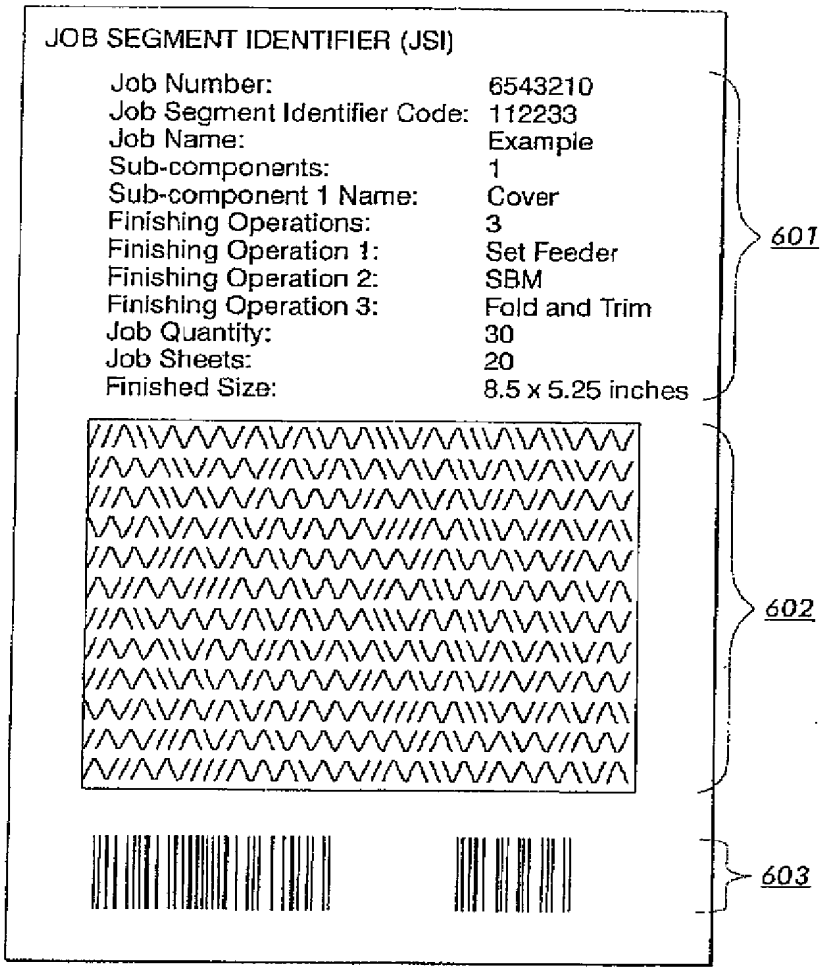

JOB SEGMENT IDENTIFIER (JSI)
Job Number: 6543210
Job Segment Identifier Code: 112233
Job Name: Example
Sub-components: 1
Sub-component 1 Name: Cover
Finishing Operations: 3
Finishing Operation 1: Set Feeder
Finishing Operation 2: SBM
Finishing Operation 3: Fold and Trim
Job Quantity: 30
Job Sheets: 20
Finished Size: 8.5 x 5.25 inches
601
602
603

FIG. 7

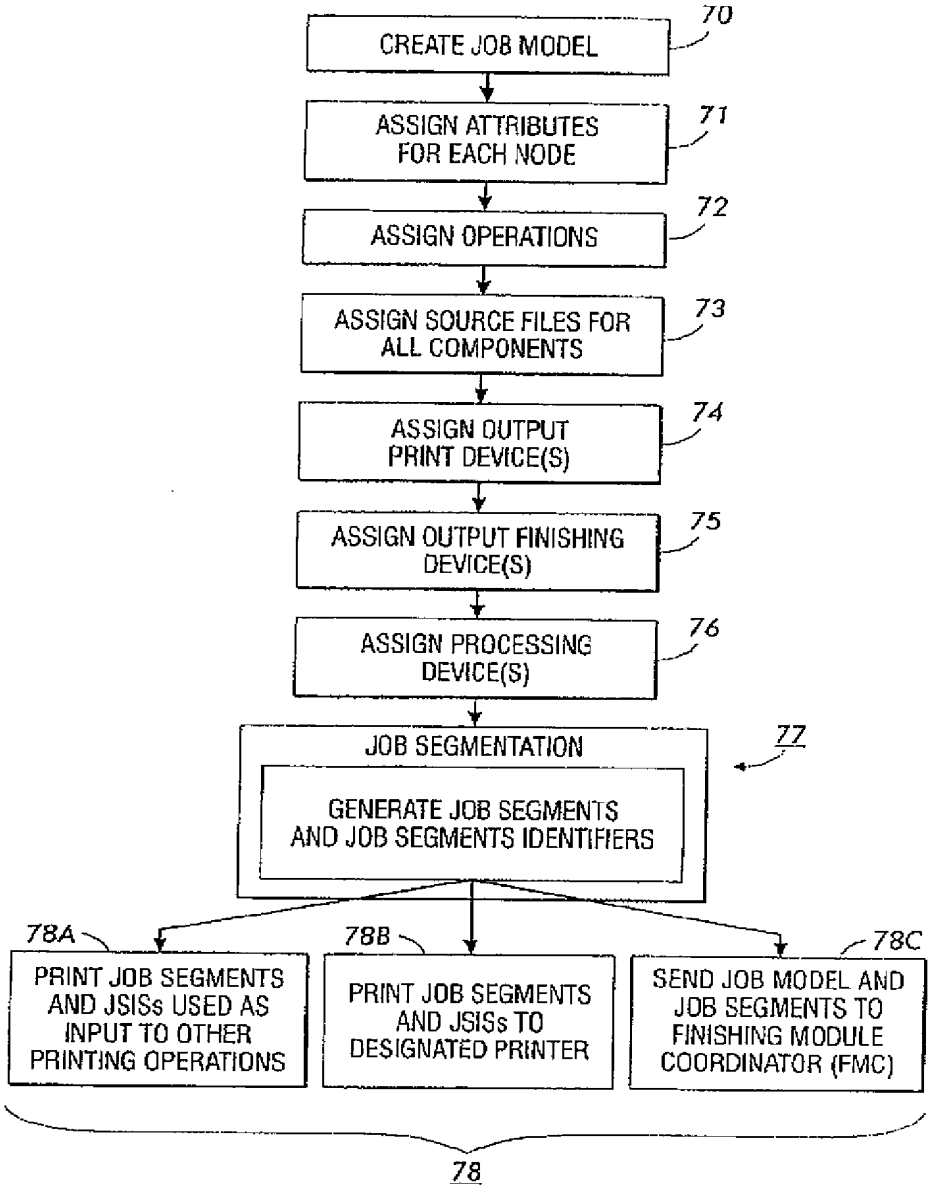

CREATE JOB MODEL
70
ASSIGN ATTRIBUTES FOR EACH NODE
71
ASSIGN OPERATIONS
72
ASSIGN SOURCE FILES FOR ALL COMPONENTS
73
ASSIGN OUTPUT PRINT DEVICE(S)
74
ASSIGN OUTPUT FINISHING DEVICE(S)
75
ASSIGN PROCESSING DEVICE(S)
76
JOB SEGMENTATION
GENERATE JOB SEGMENTS AND JOB SEGMENTS IDENTIFIERS
77
78A
PRINT JOB SEGMENTS AND JSISs USED AS INPUT TO OTHER PRINTING OPERATIONS
78B
PRINT JOB SEGMENTS AND JSISs TO DESIGNATED PRINTER
78C
SEND JOB MODEL AND JOB SEGMENTS TO FINISHING MODULE COORDINATOR (FMC)
78

FIG. 8

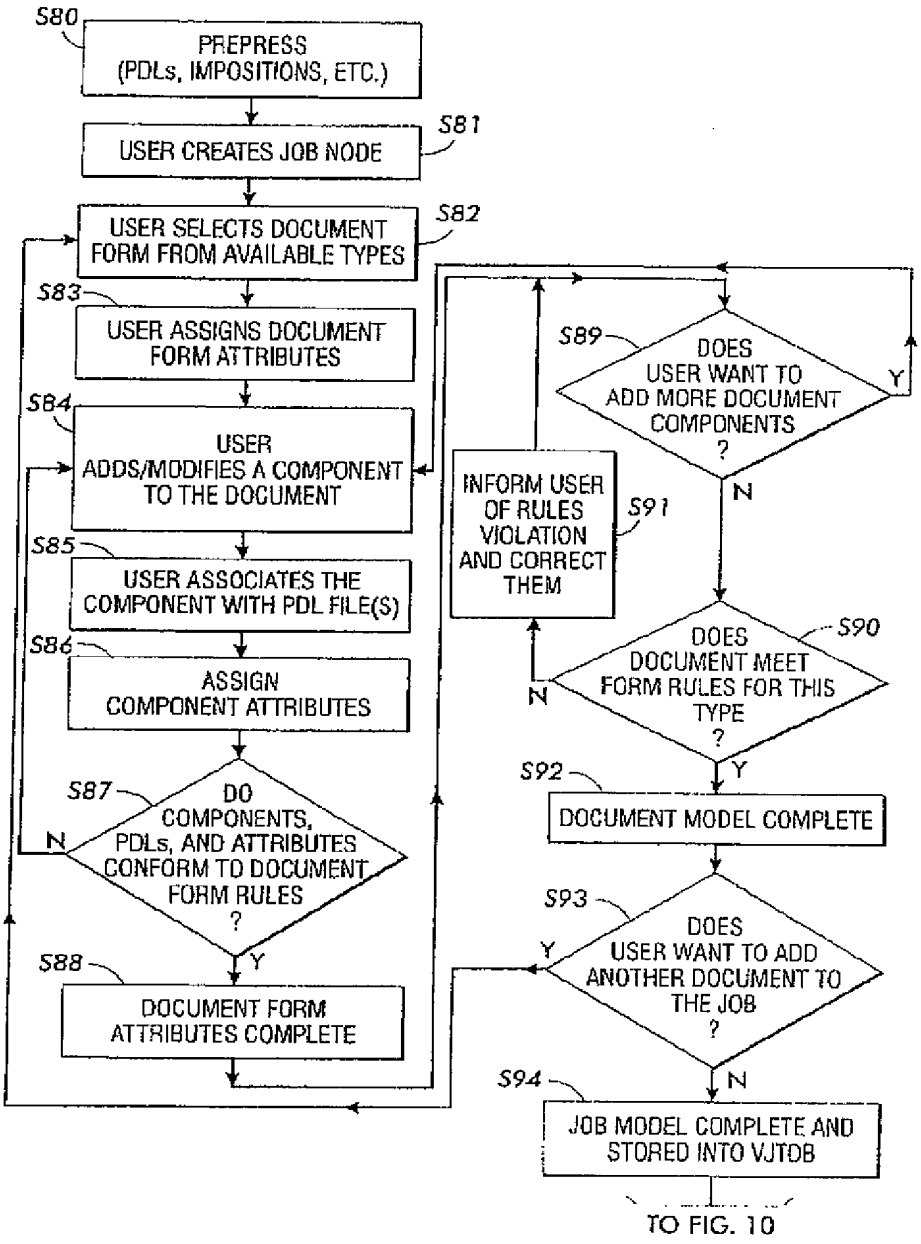
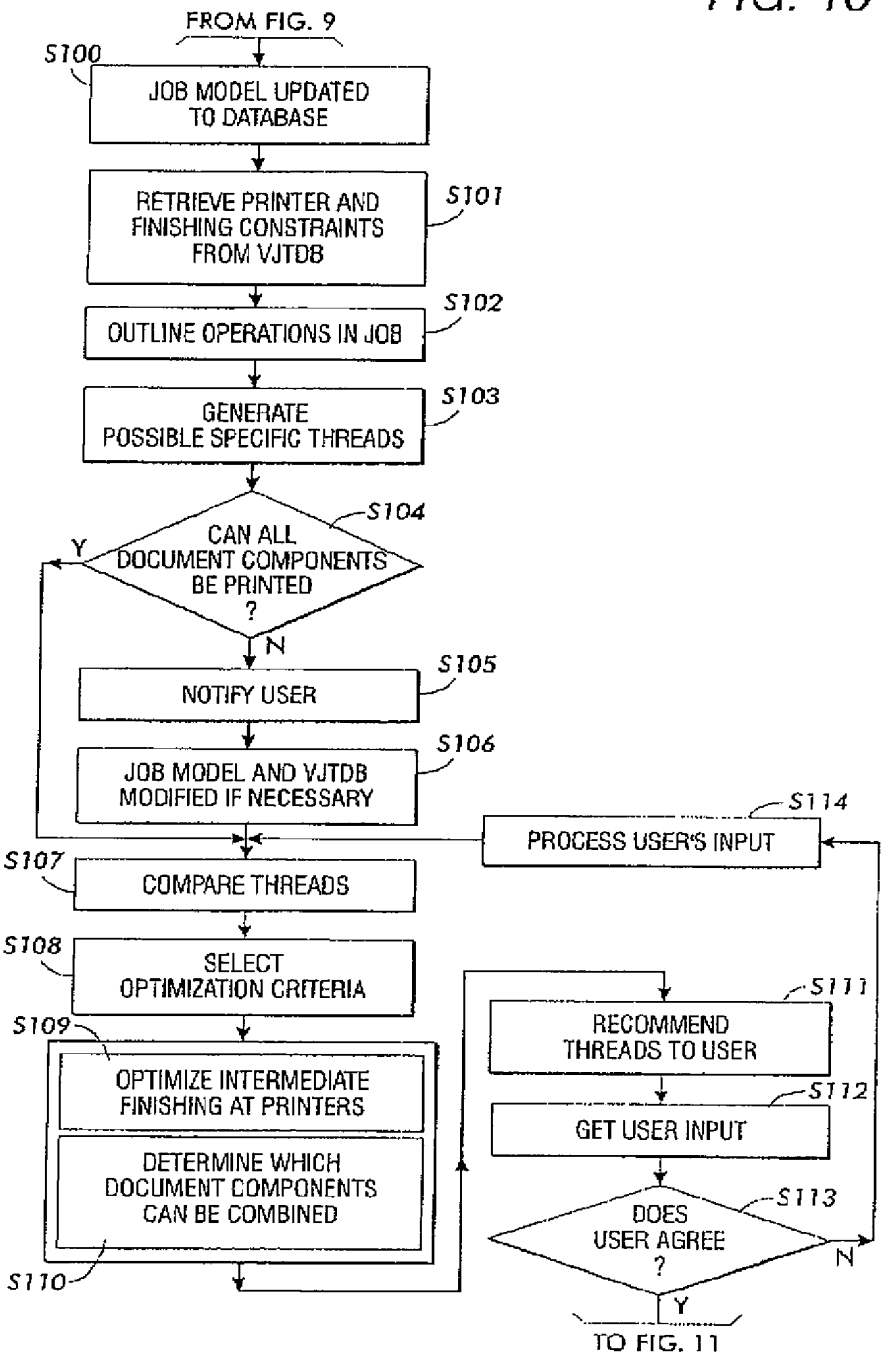

S80
PREPRESS (PDLs, IMPOSITIONS, ETC.)
S81
USER CREATES JOB NODE
S82
USER SELECTS DOCUMENT FORM FROM AVAILABLE TYPES
S83
USER ASSIGNS DOCUMENT FORM ATTRIBUTES
S84
USER ADDS/MODIFIES A COMPONENT TO THE DOCUMENT
S85
USER ASSOCIATES THE COMPONENT WITH PDL FILE(S)
S86
ASSIGN COMPONENT ATTRIBUTES
S87
DO COMPONENTS, PDLs, AND ATTRIBUTES CONFORM TO DOCUMENT FORM RULES ?
N
Y
S88
DOCUMENT FORM ATTRIBUTES COMPLETE
S89
DOES USER WANT TO ADD MORE DOCUMENT COMPONENTS ?
Y
N
S90
DOES DOCUMENT MEET FORM RULES FOR THIS TYPE ?
N
Y
S91
INFORM USER OF RULES VIOLATION AND CORRECT THEM
S92
DOCUMENT MODEL COMPLETE
S93
DOES USER WANT TO ADD ANOTHER DOCUMENT TO THE JOB ?
Y
N
S94
JOB MODEL COMPLETE AND STORED INTO VJTDB
TO FIG. 10

FIG. 9

FIG. 10


FROM FIG. 9
S100
JOB MODEL UPDATED TO DATABASE
RETRIEVE PRINTER AND FINISHING CONSTRAINTS FROM VJTDB
S101
OUTLINE OPERATIONS IN JOB
S102
GENERATE POSSIBLE SPECIFIC THREADS
S103
CAN ALL DOCUMENT COMPONENTS BE PRINTED ?
S104
Y
N
NOTIFY USER
S105
JOB MODEL AND VJTDB MODIFIED IF NECESSARY
S106
PROCESS USER'S INPUT
S114
S107
COMPARE THREADS
S108
SELECT OPTIMIZATION CRITERIA
S109
OPTIMIZE INTERMEDIATE FINISHING AT PRINTERS
DETERMINE WHICH DOCUMENT COMPONENTS CAN BE COMBINED
S110
S111
RECOMMEND THREADS TO USER
S112
GET USER INPUT
S113
DOES USER AGREE ?
N
Y
TO FIG. 11

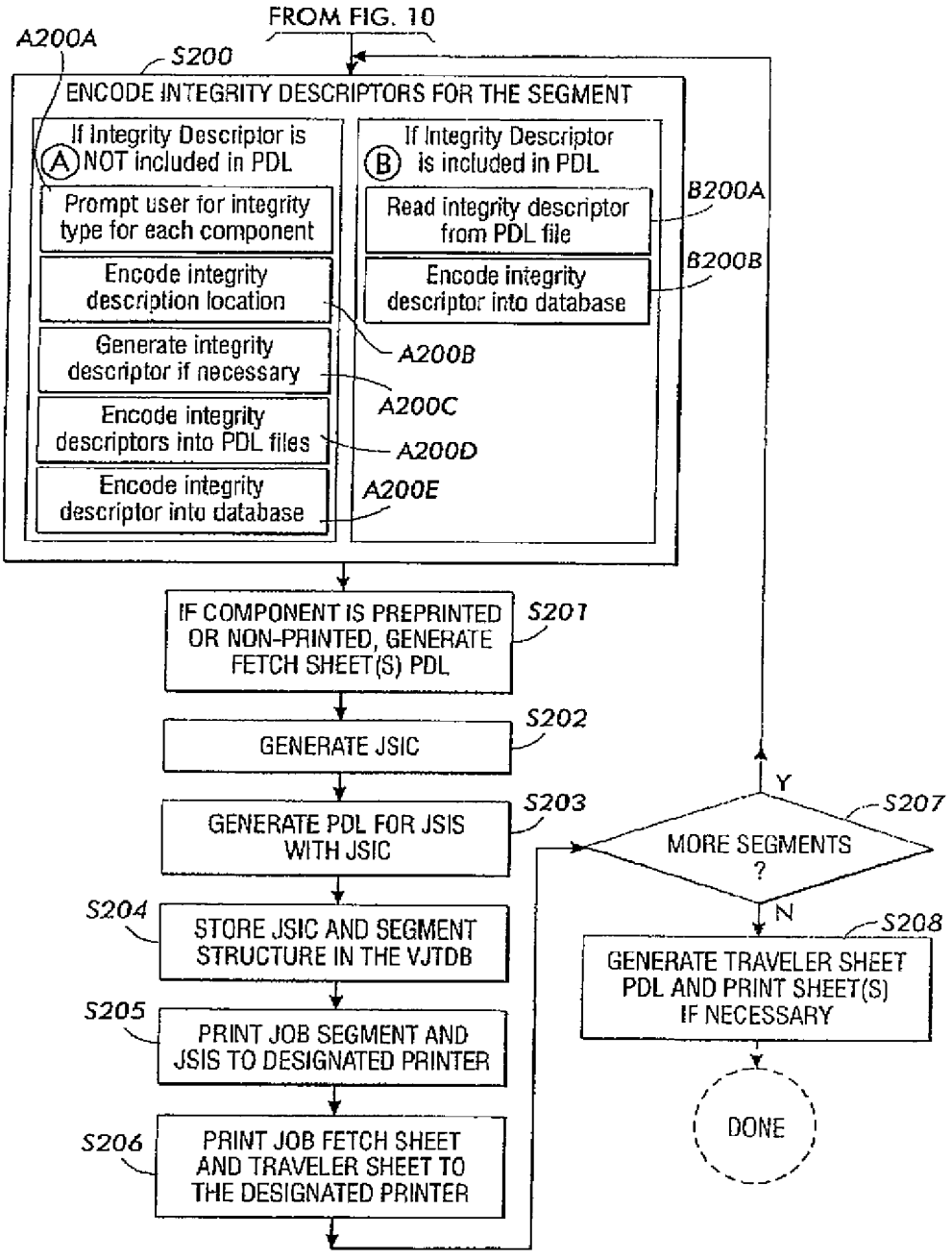

FROM FIG. 10
A200A
S200
ENCODE INTEGRITY DESCRIPTORS FOR THE SEGMENT
If Integrity Descriptor is
A NOT included in PDL
Prompt user for integrity type for each component
Encode integrity description location
Generate integrity descriptor if necessary
Encode integrity descriptors into PDL files
Encode integrity descriptor into database
A200B
A200C
A200D
A200E
If Integrity Descriptor
B is included in PDL
Read integrity descriptor from PDL file
Encode integrity descriptor into database
B200A
B200B
IF COMPONENT IS PREPRINTED OR NON-PRINTED, GENERATE FETCH SHEET(S) PDL
S201
GENERATE JSIC
S202
GENERATE PDL FOR JSIS WITH JSIC
S203
STORE JSIC AND SEGMENT STRUCTURE IN THE VJTDB
S204
PRINT JOB SEGMENT AND JSIS TO DESIGNATED PRINTER
S205
PRINT JOB FETCH SHEET AND TRAVELER SHEET TO THE DESIGNATED PRINTER
S206
MORE SEGMENTS ?
S207
Y
N
GENERATE TRAVELER SHEET PDL AND PRINT SHEET(S) IF NECESSARY
S208
DONE

FIG. 11

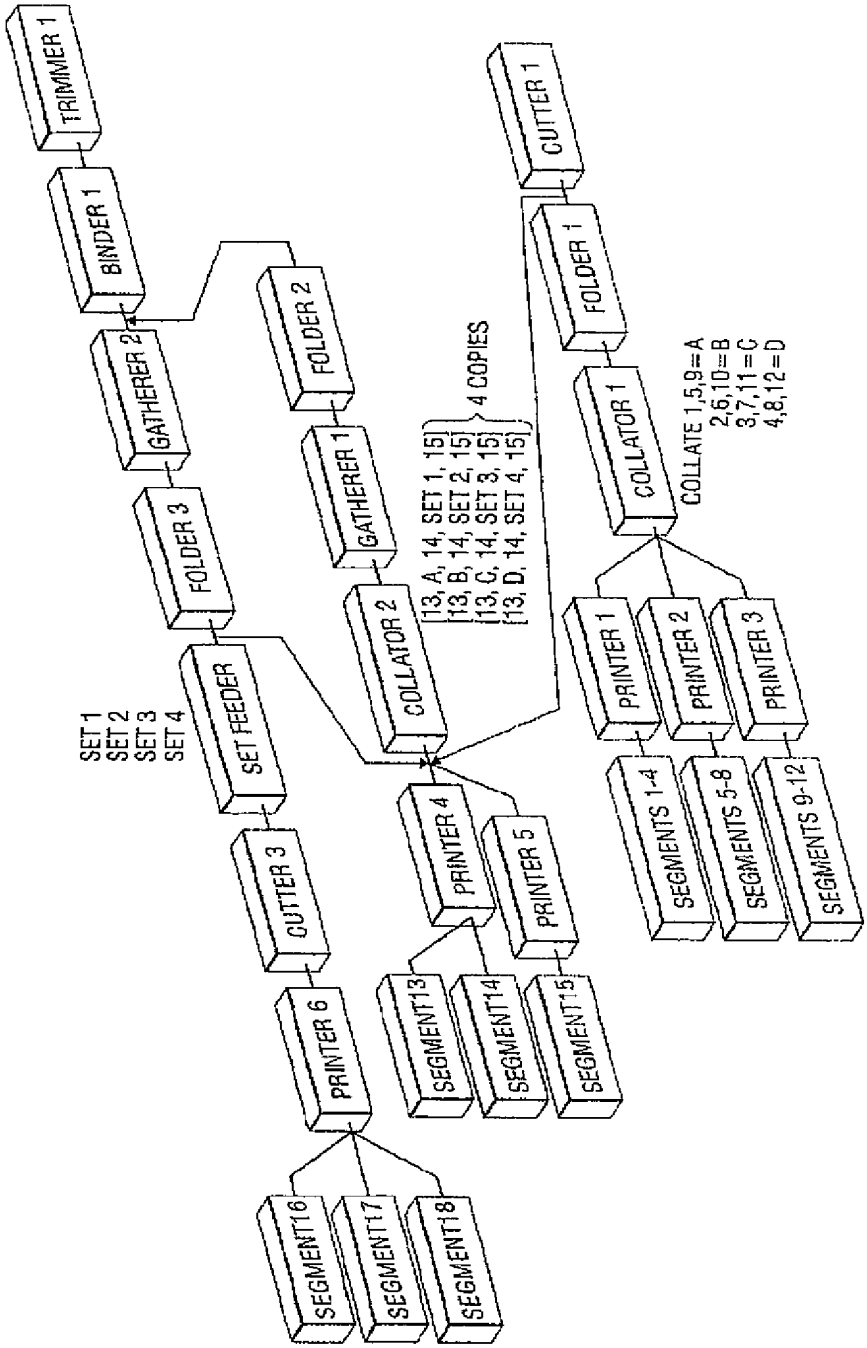

TRIMMER 1
BINDER 1
GATHERER 2
FOLDER 3
SET FEEDER
CUTTER 3
PRINTER 6
SEGMENT16
SEGMENT17
SEGMENT18
SET 1
SET 2
SET 3
SET 4
FOLDER 2
GATHERER 1
COLLATOR 2
[13, A, 14, SET 1, 15]
[13, B, 14, SET 2, 15]
[13, C, 14, SET 3, 15]
[13, D, 14, SET 4, 15]
4 COPIES
PRINTER 4
PRINTER 5
SEGMENT13
SEGMENT14
SEGMENT15
CUTTER 1
FOLDER 1
COLLATOR 1
COLLATE 1,5,9=A
2,6,10=B
3,7,11=C
4,8,12=D
PRINTER 1
PRINTER 2
PRINTER 3
SEGMENTS 1-4
SEGMENTS 5-8
SEGMENTS 9-12

FIG. 12

FIG. 13

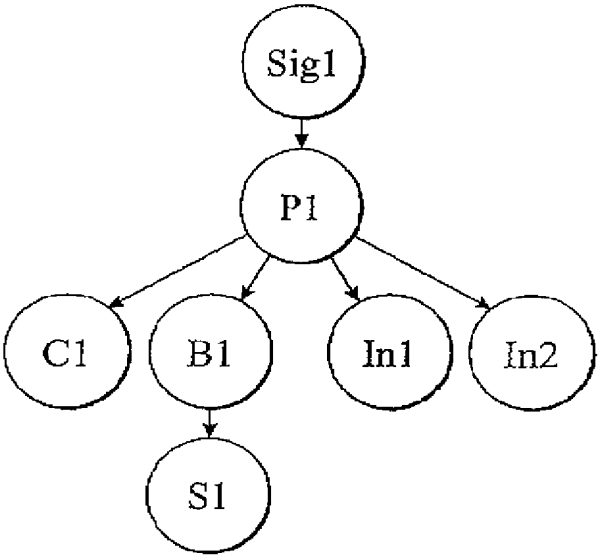
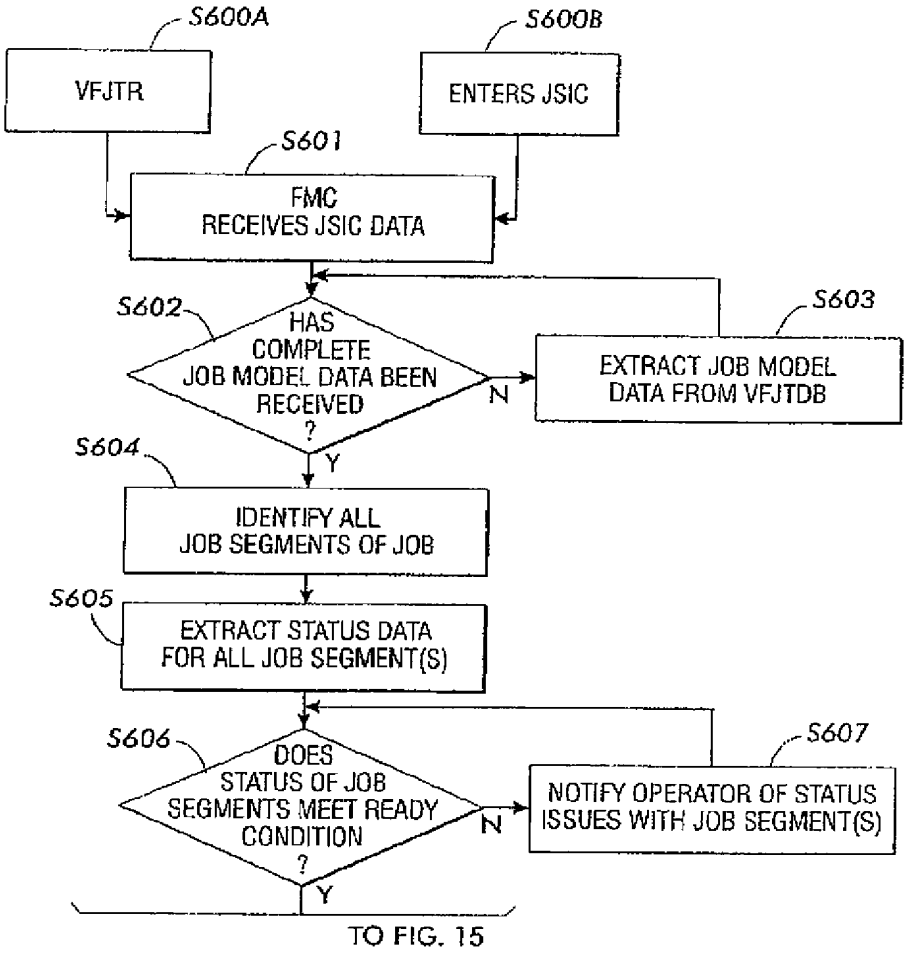


FIG. 14


S600A
VFJTR
S600B
ENTERS JSIC
S601
FMC
RECEIVES JSIC DATA
S602
HAS
COMPLETE
JOB MODEL DATA BEEN
RECEIVED
?
N
S603
EXTRACT JOB MODEL
DATA FROM VFJTDB
Y
S604
IDENTIFY ALL
JOB SEGMENTS OF JOB
S605
EXTRACT STATUS DATA
FOR ALL JOB SEGMENT(S)
S606
DOES
STATUS OF JOB
SEGMENTS MEET READY
CONDITION
?
N
S607
NOTIFY OPERATOR OF STATUS
ISSUES WITH JOB SEGMENT(S)
Y
TO FIG. 15

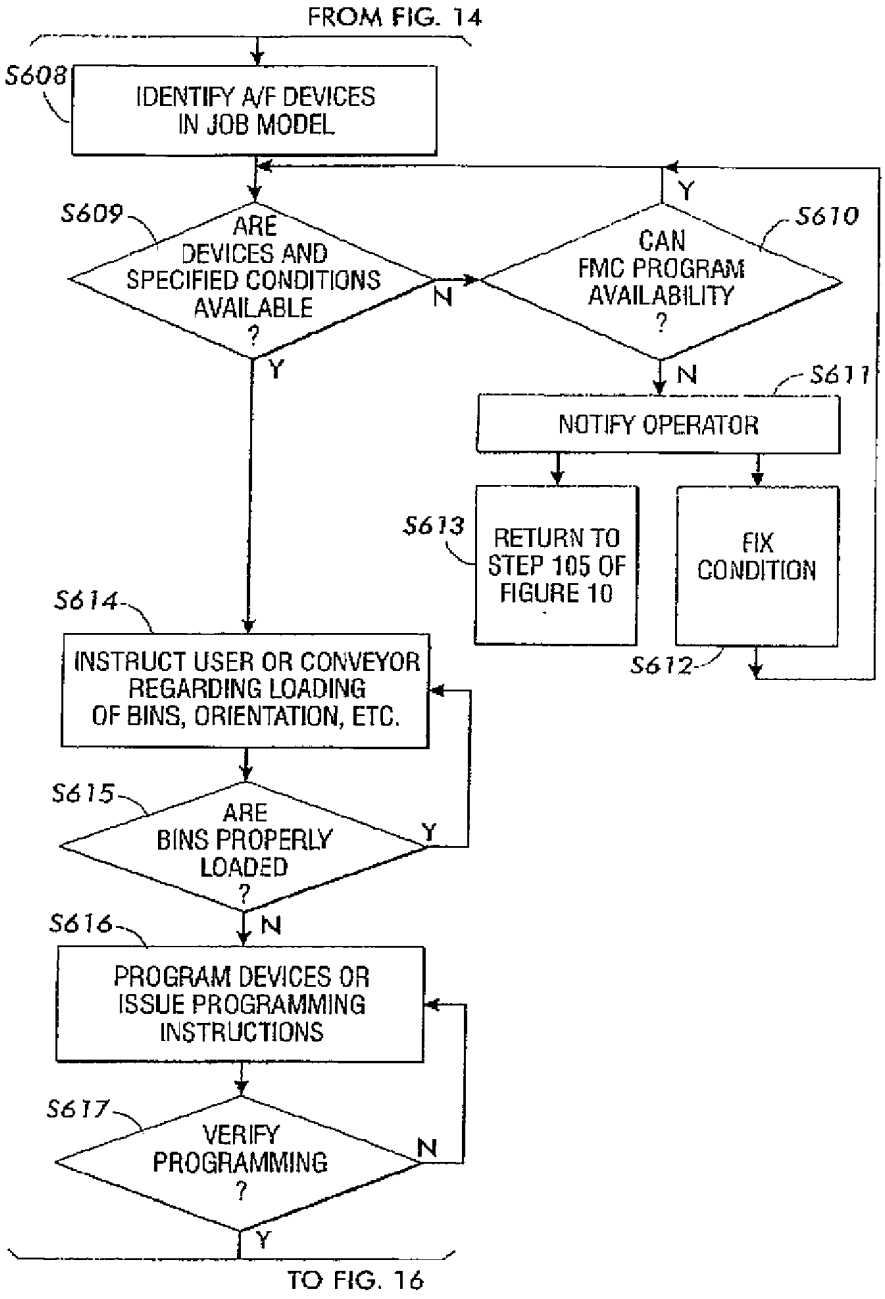

FROM FIG. 14
S608
IDENTIFY A/F DEVICES IN JOB MODEL
S609
ARE DEVICES AND SPECIFIED CONDITIONS AVAILABLE ?
N
Y
S610
CAN FMC PROGRAM AVAILABILITY ?
Y
N
S611
NOTIFY OPERATOR
S613
RETURN TO STEP 105 OF FIGURE 10
FIX CONDITION
S612
S614
INSTRUCT USER OR CONVEYOR REGARDING LOADING OF BINS, ORIENTATION, ETC.
S615
ARE BINS PROPERLY LOADED ?
Y
N
S616
PROGRAM DEVICES OR ISSUE PROGRAMMING INSTRUCTIONS
S617
VERIFY PROGRAMMING ?
N
Y
TO FIG. 16

FIG. 15

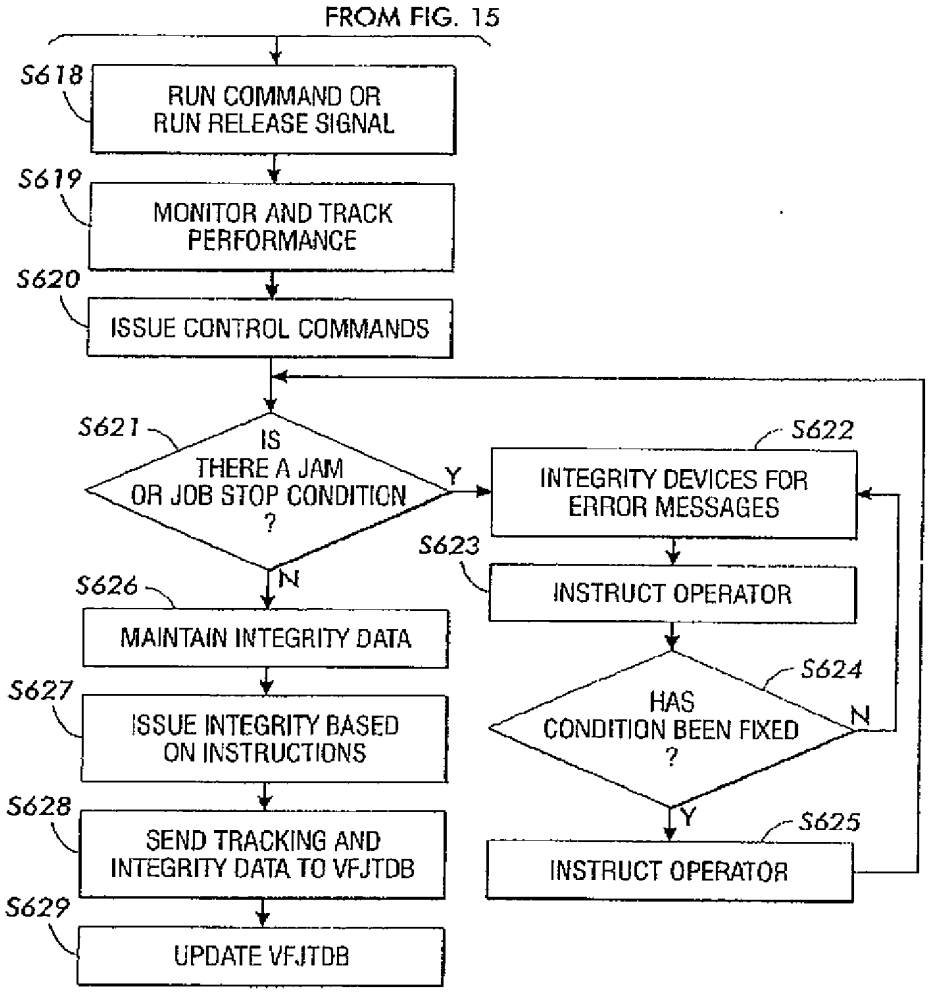

FROM FIG. 15
S618
RUN COMMAND OR RUN RELEASE SIGNAL
S619
MONITOR AND TRACK PERFORMANCE
S620
ISSUE CONTROL COMMANDS
S621
IS THERE A JAM OR JOB STOP CONDITION ?
Y
N
S622
INTEGRITY DEVICES FOR ERROR MESSAGES
S623
INSTRUCT OPERATOR
S624
HAS CONDITION BEEN FIXED ?
N
Y
S625
INSTRUCT OPERATOR
S626
MAINTAIN INTEGRITY DATA
S627
ISSUE INTEGRITY BASED ON INSTRUCTIONS
S628
SEND TRACKING AND INTEGRITY DATA TO VFJTDB
S629
UPDATE VFJTDB

FIG. 16

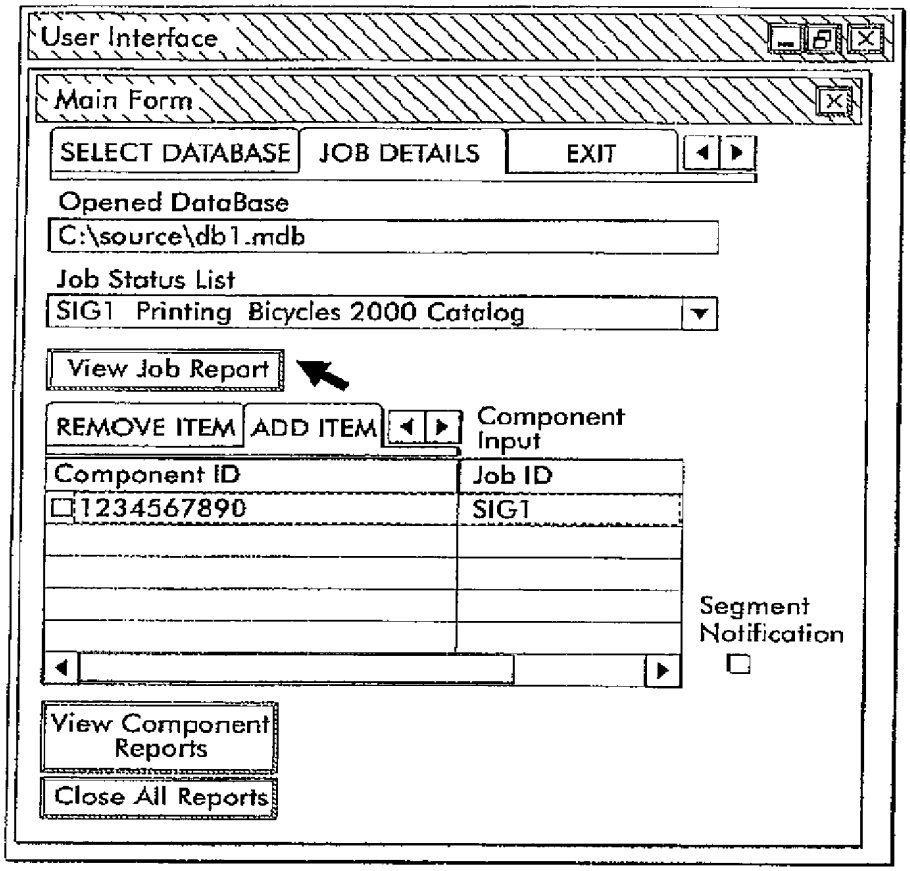

User Interface
Main Form
SELECT DATABASE
JOB DETAILS
EXIT
Opened DataBase
C:\source\db1.mdb
Job Status List
SIG1 Printing Bicycles 2000 Catalog
View Job Report
REMOVE ITEM
ADD ITEM
Component Input
Component ID
Job ID
1234567890
SIG1
Segment Notification
View Component Reports
Close All Reports

FIG. 17

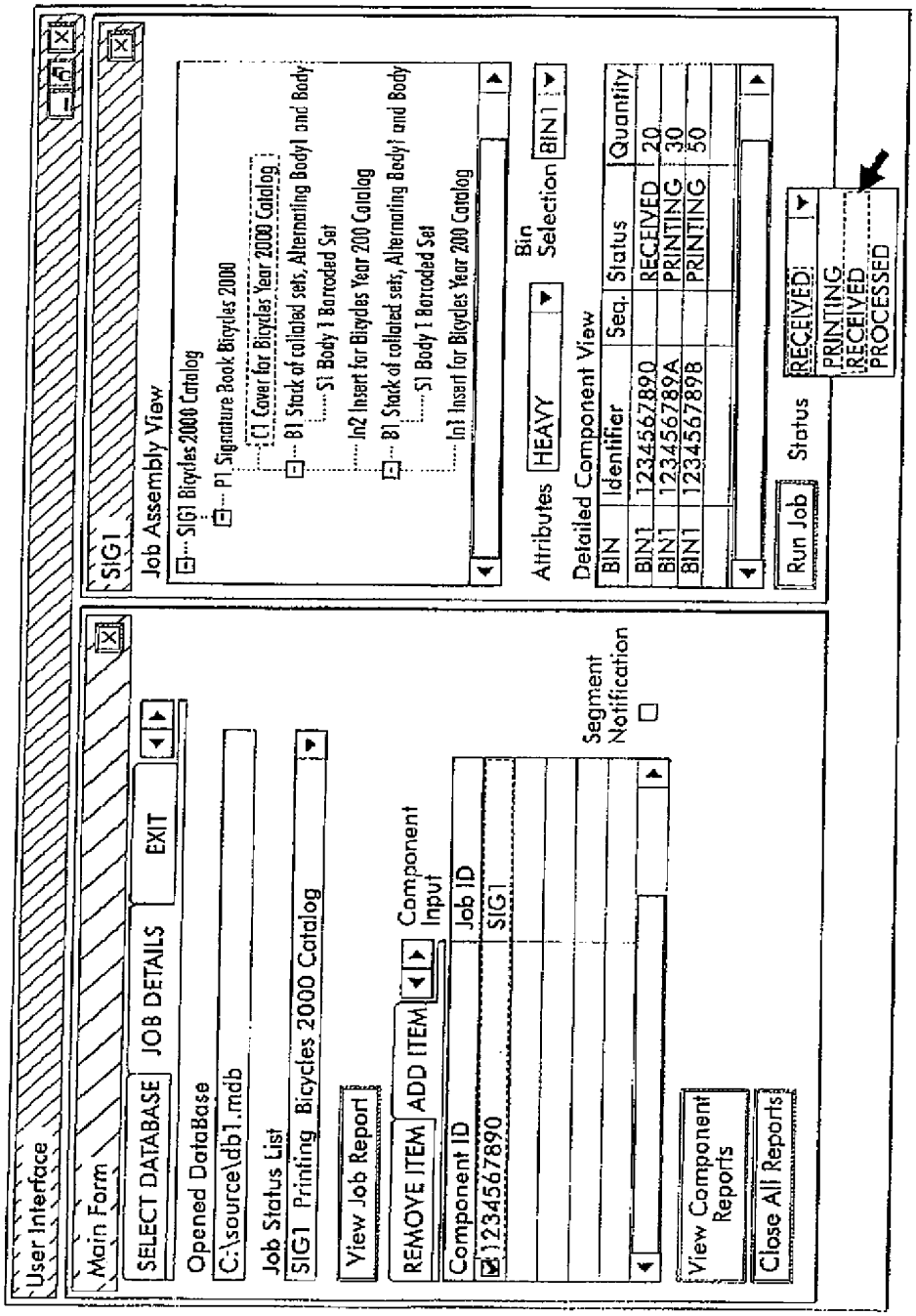

User Interface
Main Form
SELECT DATABASE
JOB DETAILS
EXIT
Opened DataBase
C:\source\db1.mdb
Job Status List
SIG1 Printing Bicycles 2000 Catalog
View Job Report
REMOVE ITEM
ADD ITEM
Component Input
Component ID
Job ID
1234567890
SIG1
Segment Notification
View Component Reports
Close All Reports
SIG1
Job Assembly View
SIG1 Bicycles 2000 Catalog
P1 Signature Book Bicycles 2000
C1 Cover for Bicycles Year 2000 Catalog
B1 Stack of collated sets, Alternating Body1 and Body
S1 Body 1 Barcoded Set
In2 Insert for Bicycles Year 200 Catalog
B1 Stack of collated sets, Alternating Body1 and Body
S1 Body 1 Barcoded Set
In1 Insert for Bicycles Year 200 Catalog
Attributes
HEAVY
Bin Selection
BIN1
Detailed Component View
BIN
Identifier
Seq.
Status
Quantity
BIN1 1234567890 RECEIVED 20
BIN1 1234567892A PRINTING 30
BIN1 1234567898 PRINTING 50
Run Job
Status
RECEIVED
PRINTING
RECEIVED
PROCESSED

FIG. 18

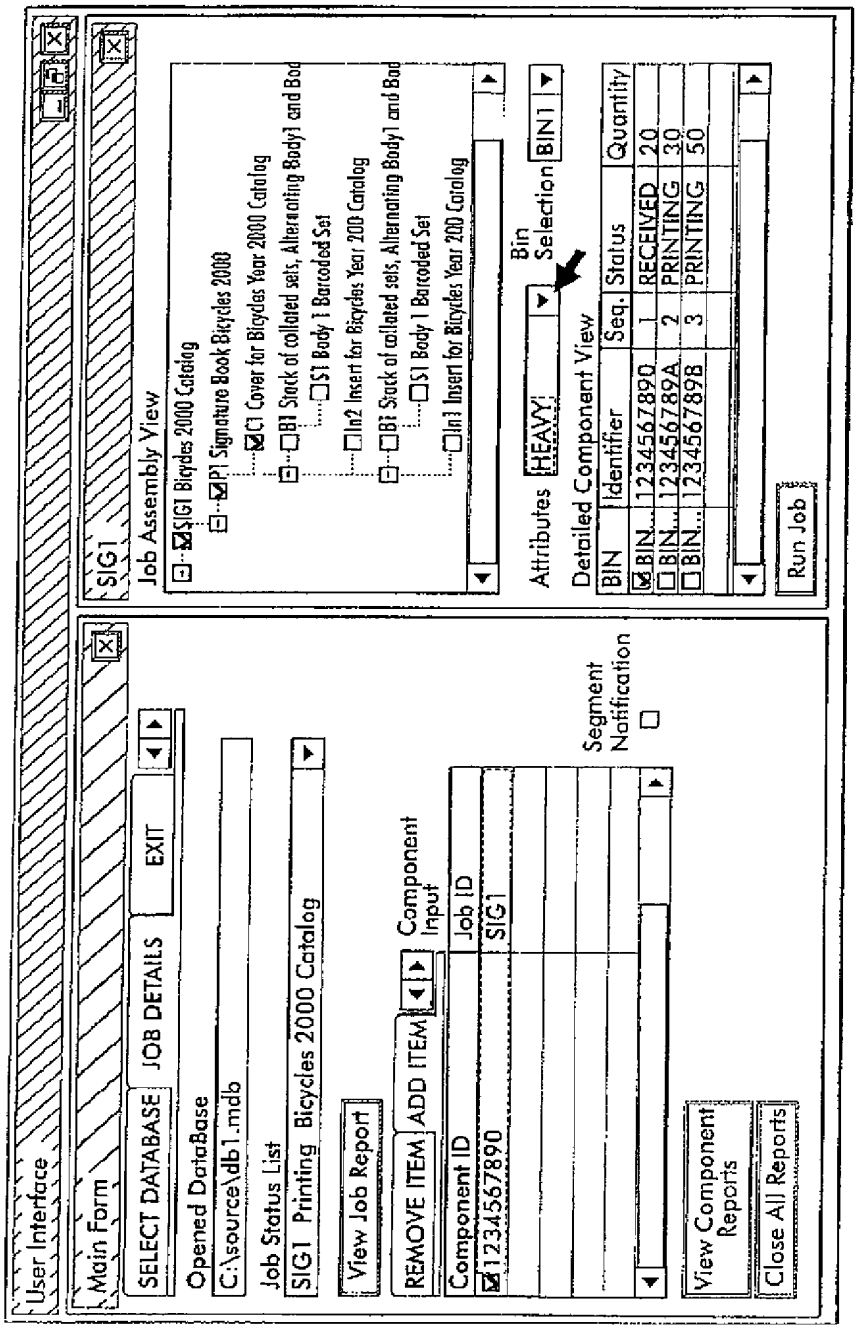

User Interface
Main Form
SELECT DATABASE
JOB DETAILS
EXIT
Opened DataBase
C:\source\db1.mdb
Job Status List
SIG1 Printing Bicycles 2000 Catalog
View Job Report
REMOVE ITEM
ADD ITEM
Component Input
Component ID
Job ID
1234567890
SIG1
Segment Notification
View Component Reports
Close All Reports
SIG1
Job Assembly View
$IG1 Bicycles 2000 Catalog
P1 Signature Book Bicycles 2000
C1 Cover for Bicycles Year 2000 Catalog
B1 Stack of collated sets, Alternating Body1 and Bod
S1 Body 1 Barcoded Set
In2 Insert for Bicycles Year 200 Catalog
B1 Stack of collated sets, Alternating Body1 and Bod
S1 Body 1 Barcoded Set
In3 Insert for Bicycles Year 200 Catalog
Attributes
HEAVY
Bin Selection
BIN1
Detailed Component View
BIN
Identifier
Seq.
Status
Quantity
BIN... 1234567890 1 RECEIVED 20
BIN... 1234567891A 2 PRINTING 30
BIN... 1234567891B 3 PRINTING 50
Run Job

FIG. 19

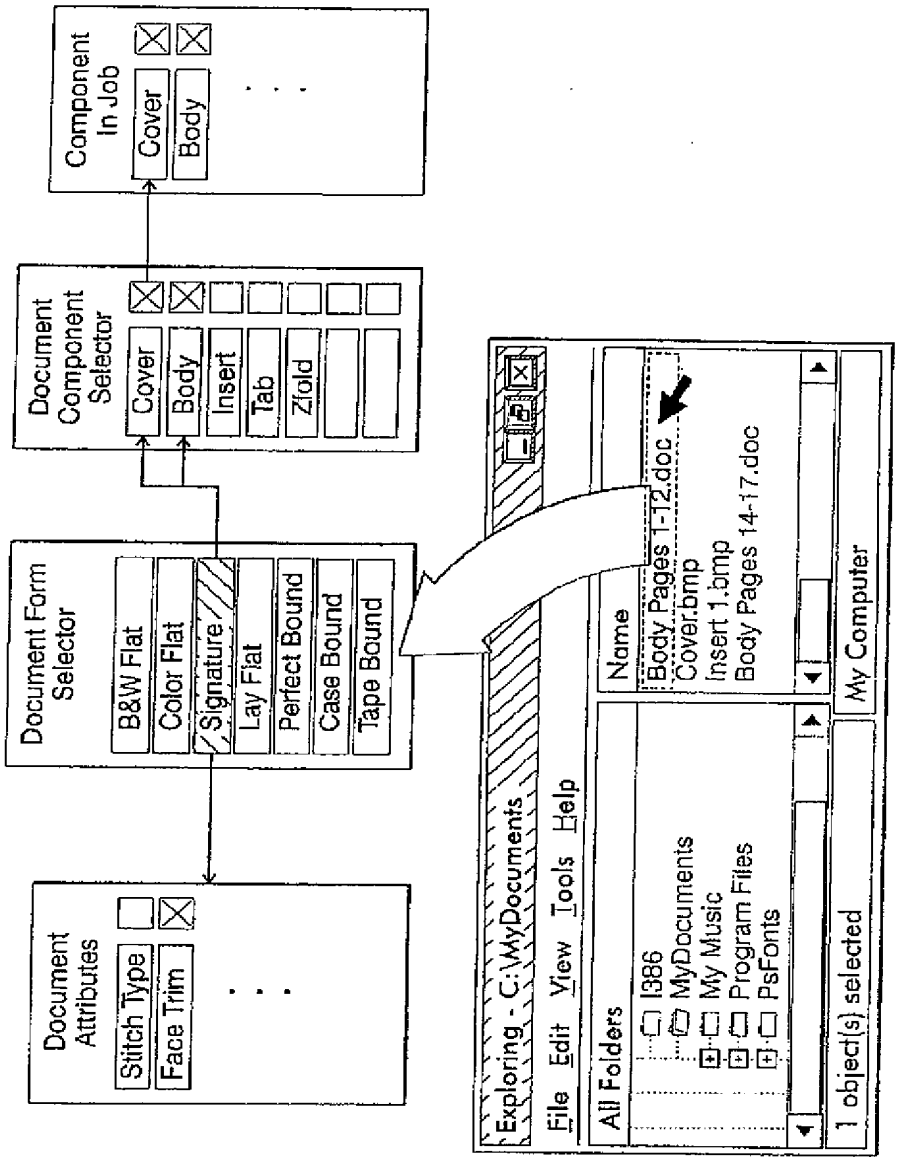

Component In Job
Cover
Body
Document Component Selector
Cover
Body
Insert
Tab
Zfold
Document Form Selector
B&W Flat
Color Flat
Signature
Lay Flat
Perfect Bound
Case Bound
Tape Bound
Document Attributes
Stitch Type
Face Trim
Exploring - C:\MyDocuments
File Edit View Tools Help
All Folders
I386
MyDocuments
My Music
Program Files
PsFonts
Name
Body Pages 1-12.doc
Cover.bmp
Insert 1.bmp
Body Pages 14-17.doc
My Computer
1 object(s) selected

FIG. 20

[DOCUMENT NAME] ABSTRACT OF THE DISCLOSURE

[SUMMARY]

[PROBLEM TO BE SOLVED]

To provide a system for electronic management and control of a wide range of finishing processes characterized by input from multiple production operations and equipment that may be variably applied to work pieces that themselves are highly variable between different jobs.

[MEANS TO SOLVE THE PROBLEM]

A job model file is created (70), attributes are assigned to each node identified in the high level job model (71), and the operation to be performed on each mode are identified (72). Source files for each of the document components are assigned (73), printers are assigned (74), output finishing devices are assigned (75), and processing devices are assigned (76). Job segments for operation of the job are determined based upon the attributes and operations associated with each document component (77). Various outputs are shown (78A-78C).

[REPRESENTATIVE DRAWING]

Fig. 8